University of Alberta

New Representations and Approximations for Sequential Decision Making under Uncertainty

by

Tao Wang

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

Edmonton, Alberta
Fall 2007

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-33088-3
Our file Notre référence
ISBN: 978-0-494-33088-3

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

This dissertation research addresses the challenge of scaling up algorithms for sequential decision making under uncertainty. In my dissertation, I developed new approximation strategies for planning and learning in the presence of uncertainty while maintaining useful theoretical properties that allow larger problems to be tackled than is practical with exact methods. In particular, my research tackles three outstanding issues in sequential decision making in uncertain environments: performing stable generalization during off-policy updates, balancing exploration with exploitation, and handling partial observability of the environment.

The first key contribution of my thesis is the development of novel dual representations and algorithms for planning and learning in stochastic environments. This dual view I have developed offers a coherent and comprehensive approach to optimal sequential decision making problems, provides an alternative to standard value function based techniques, and opens new avenues for solving sequential decision making problems. In particular, I have shown that dual dynamic programming algorithms can avoid the divergence problems associated with the standard primal approach, even in the presence of approximation and off-policy updates.

Another key contribution of my thesis is the development of a practical action selection strategy that addresses the well known exploration versus exploitation tradeoff in reinforcement learning. The idea is to exploit information in a Bayesian posterior to make intelligent actions by growing an adaptive, sparse lookahead tree. This technique evaluates actions while taking into account any effects they might have on future knowledge, as well as future reward, and outperforms current selection strategies.

Finally my thesis also develops a new approach to approximate planning in

partially observable Markov decision processes. Here the challenge is to overcome the exponential space required by standard value iteration. For this problem, I introduced a new, quadratic upper bound approximation that can be optimized by semidefinite programming. This approach achieves competitive approximation quality while maintaining a compact representation; requiring computational time and space that is only linear in the number of decisions.

Overall, my dissertation research developed new tools for computing optimal sequential decision strategies in stochastic environments, and has contributed significant progress on three key challenges in reinforcement learning.

Acknowledgements

I would like to thank my advisors, Dale Schuurmans and Michael Bowling, for their guidance, encouragement, many discussions and other support. Without their help, this dissertation would not have happened. I also would like to thank my committee: Richard S. Sutton, Paul R. Messinger, and Doina Precup for their helpful comments and suggestions.

Table of Contents

1 Introduction **1**
- 1.1 Contributions . . . 3
- 1.2 Thesis Outline . . . 4

2 Background: Markov Decision Processes **5**
- 2.1 Optimality Criteria for MDPs . . . 6
- 2.2 Solution Techniques for MDPs . . . 9
  - 2.2.1 Planning Algorithms . . . 9
  - 2.2.2 Learning Algorithms . . . 11
  - 2.2.3 Hybrid Solution Techniques for MDPs . . . 14
- 2.3 Approximate Solution Techniques for MDPs . . . 14
  - 2.3.1 Value Function Approximation . . . 15
  - 2.3.2 Sampling . . . 16
  - 2.3.3 Exploiting Model Structure . . . 17
- 2.4 Partially Observable Markov Decision Processes . . . 19
- 2.5 Summary . . . 20

3 Dual Representations and Dual Algorithms **22**
- 3.1 Motivation . . . 23
- 3.2 Preliminaries . . . 24
- 3.3 Linear Programming . . . 25
- 3.4 On-Policy Update . . . 29
  - 3.4.1 State Based Policy Evaluation . . . 29
  - 3.4.2 State-Action Based Policy Evaluation . . . 35
  - 3.4.3 Convergence Analysis . . . 40
- 3.5 Policy Improvement . . . 43
- 3.6 Off-Policy Update . . . 46
  - 3.6.1 Primal Representations . . . 46
  - 3.6.2 Dual Representations . . . 47
  - 3.6.3 Convergence Analysis . . . 48
- 3.7 Approximate Dynamic Programming . . . 51
  - 3.7.1 Approximate Representations . . . 51
  - 3.7.2 Projection Operator . . . 54
  - 3.7.3 Gradient Operator . . . 59
- 3.8 Experimental Results . . . 63
  - 3.8.1 Task: Randomly Synthesized MDPs . . . 65
  - 3.8.2 Task: The Star Problem . . . 65
  - 3.8.3 Task: The Mountain Car Problem . . . 66
- 3.9 Complexity Analysis of Dual Algorithms . . . 67
- 3.10 Conclusion . . . 71

4 A Sparse Sampling Approach to Action Selection **73**
4.1 Motivation . . . 74
4.2 Bayesian Statistics . . . 75
4.3 Bayesian Reinforcement Learning . . . 77
4.4 Action Selection . . . 80
4.4.1 Non-Bayesian Methods for Action Selection . . . 81
4.4.2 Bayesian Action Selection . . . 83
4.5 Bayesian Sparse Sampling . . . 87
4.6 Experimental Results . . . 91
4.7 Conclusion . . . 95

5 Quadratic Approximations for POMDP Planning **97**
5.1 Motivation . . . 98
5.2 Partially Observable Markov Decision Processes . . . 100
5.3 Value Function Approximation Strategies . . . 103
5.4 Convex Quadratic Upper Bounds . . . 104
5.4.1 Convex Upper Bound Iteration . . . 106
5.4.2 Algorithmic Approach . . . 108
5.5 Experimental Results . . . 110
5.6 Conclusion . . . 111

6 Conclusions **114**
6.1 Contributions . . . 114
6.2 Future Research . . . 116

Bibliography **118**

A Notation for Chapter 3 **126**

B Tabular Reinforcement Learning **127**
B.1 Temporal Difference Evaluation . . . 127
B.2 Sarsa: On-Policy TD Control . . . 128
B.3 Q-Learning: Off-Policy TD Control . . . 128

C Approximate Reinforcement Learning **130**

D Common Probability Distributions **135**

E Some Useful Conjugate Priors **136**

F Other Related Work **137**
F.1 Exponential Family . . . 137
F.2 Multi-armed Bandit Problem . . . 142
F.3 Batch Methods for Action Selection . . . 143

List of Tables

2.1 The policy iteration algorithm 12
2.2 The value iteration algorithm 12
2.3 The Q-learning algorithm 13
2.4 The Sarsa algorithm 13
2.5 Sketch of the sparse sampling algorithm 18

3.1 Computational complexity of the algorithms 67

4.1 On-line methods for action selection 81
4.2 A Bayesian sparse sampling algorithm 90

5.1 Problem characteristics 112
5.2 Experimental results 112

D.1 Some common probability distributions 135

E.1 Some useful conjugate priors 136

List of Figures

2.1 Illustration of a lookahead tree 18

3.1 The star problem 68
3.2 On-policy update of state-action value q and visit distribution H on randomly synthesized MDPs 68
3.3 Off-policy update of state-action value q and visit distribution H on randomly synthesized MDPs 69
3.4 On-policy update of state-action value q and visit distribution H on the star problem 69
3.5 Off-policy update of state-action value q and visit distribution H on the star problem 70
3.6 On-policy update of state-action value q and visit distribution H on the mountain car problem 70
3.7 Off-policy update of state-action value q and visit distribution H on the mountain car problem 72

4.1 Illustration of an irregular lookahead tree 90
4.2 Results: Bernoulli bandits 93
4.3 Results: Gaussian bandits 93
4.4 Results: 1-dimensional continuous action Gaussian process 94
4.5 Results: 2-dimensional continuous action Gaussian process 94

5.1 Illustration of a convex quadratic upper bound approximation to a maximum of linear functions $b \cdot \alpha_\pi$. 105

Chapter 1

Introduction

Imagine a mobile robot selling candies, bustling around the building to optimize its profit. The robot must decide where to visit while navigates from one place to another. The outcomes of the robot's actions are not totally predictable because its environment is uncertain. The robot is facing a problem of sequential decision making under uncertainty. At each decision point, the robot has to choose an action. After executing the action, it finds itself in a new state while receiving feedback or reward from the environment (e.g., someone buying a candy). The goal of the robot is to choose a sequence of actions to affect its environment so that it can maximize its total reward.

The solution to a decision making problem is called a policy: a behavior strategy for selecting actions based on the states of the environment. Therefore, solving a decision making problem amounts to computing an optimal policy. The sequential decision making problem can be categorized as either a *planning problem*, where one is given a complete specification of the environment dynamics and reward function (i.e., a model), or a *learning problem* where one does not initially know the model. The planning problem is normally but not exclusively tackled by *linear programming* (LP) or *dynamic programming* (DP) methods, whereas the learning problem is solved by *reinforcement learning* (RL) methods. Since RL techniques generalize DP techniques, they can also be applied when the environment is known.

My research attempts to tackle three key outstanding issues in sequential decision making under uncertainty: combining off-policy updates with generalization, balancing exploration and exploitation, and handling partial observability of the

environment.

First, off-policy learning means learning about a policy other than the one that is currently being followed by the agent. Off-policy learning is appealing for two reasons: (1) this mechanism allows learning about many different policies in parallel with the same stream of experience from following a single policy (e.g., learning to walk to a meeting room from the experience of moving to a lab); (2) it allows for learning the value of an optimal policy by following a suboptimal policy (e.g., Q-learning), which is very useful when the optimal policy is unknown. Unfortunately, it is well-known that Q-learning with linear function approximation (a way to generalize the state space) can diverge. It is challenging but desirable to gain better convergence guarantees for off-policy updates with generalization.

Second, the problem of balancing exploration and exploitation arises when the model of an environment is unknown. At each decision point, the robot faces the challenging issue of how to choose its action during learning. Here there is a fundamental tradeoff: it can exploit its current knowledge to gain reward by taking actions known to give relatively high reward, or explore the environment to gain information by trying actions whose value is uncertain. This is the famous problem of trading off between exploitation and exploration during action selection in reinforcement learning.

Third, partial observability of the environment is another important issue in decision making. One commonly used formal specifications of sequential decision making problems is the *partially observable Markov decision process* (POMDP), which does not assume full observability of the state of the environment. Unfortunately, even approximate planning in POMDPs is known to be hard, and developing heuristic planners that can deliver reasonable results in practice has proved to be a significant challenge.

This thesis seeks to address these three key outstanding problems in sequential decision making under uncertainty from the perspective of representation, planning, and learning. In particular, I am trying to answer the following questions:

- Is it possible to do off-policy updates with generalization while avoiding the risk of divergence? If possible, how?

- How can we balance the fundamental trade-off between exploration and exploitation during action selection?
- How can we approximate POMDP planning efficiently in terms of memory and computation?

1.1 Contributions

Addressing the above three problems, this thesis pursues new approximation strategies for planning and learning under uncertainty that maintain useful theoretical properties, while allowing larger problems to be tackled than is practical with exact methods. The key contributions of this thesis are three-fold.

Dual representations and algorithms (Chapter 3). I present dual representations and a body of novel dual algorithms for planning and learning in sequential decision making environments. I also investigate the convergence properties of these new algorithms both theoretically and empirically. Dual representations maintain an explicit representation of stationary distributions as opposed to value functions. Therefore, dual update algorithms, since they are based on estimating normalized probability distributions rather than unbounded value functions, avoid divergence even in the presence of approximation and off-policy updates. Moreover, this dual view offers a coherent and comprehensive perspective on optimal sequential decision making problems, provides a viable alternative to standard value function based techniques, and opens new avenues for solving sequential decision making problems.

A new technique for on-line action selection (Chapter 4). I propose a relatively straightforward action selection strategy to address the well known exploration versus exploitation tradeoff. The technique exploits information in a Bayesian posterior to make intelligent actions by growing an adaptive, sparse lookahead tree. I demonstrate that Bayesian sparse sampling improves action selection in simple reinforcement learning scenarios.

A novel algorithm for approximate POMDP planning (Chapter 5). I introduce a new approach to approximate value-iteration for POMDP planning that is based on quadratic rather than piecewise linear function approximators. Specifically, I approximate the optimal value function by a convex upper bound composed of a fixed number of quadratics, and optimize it by semidefinite programming. I demonstrate that my approach can achieve competitive approximation quality to current techniques while still maintaining a bounded size representation of the value function approximator. Overall, the technique requires computation time and space that is only linear in the number of decision stages.

1.2 Thesis Outline

The thesis is organized as follows. Chapter 2 presents the standard background on Markov Decision Processes (MDPs) and basic reinforcement learning algorithms. Chapter 3 introduces novel dual representations, new forms of dynamic programming algorithms and their convergence analysis. Chapter 4 presents the idea of Bayesian sparse sampling, a new on-line action selection strategy that grows a sparse lookadhead tree. Chapter 5 introduces an approximate algorithm for POMDP planning, which represents value functions as quadratics and optimizes them by semidefinite programming. Chapter 6 concludes with a summary of my contributions and a consideration of future research.

Publication Notes

Some of the material from Chapter 3 appears in (Wang et al., 2007). The key results from Chapter 4 were reported in (Wang et al., 2005). The work in Chapter 5 was published in (Wang et al., 2006).

Chapter 2

Background: Markov Decision Processes

In this chapter, I present some fundamental background on Markov decision processes. I then conclude this chapter with a list of three key outstanding problems in sequential decision making in uncertain environments, which will be addressed in the following chapters of the thesis. The goal here is to establish the foundation for the presentation of my thesis research. A more comprehensive treatment of Markov decision processes can be found in standard texts (Puterman, 1994; Bertsekas, 1995).

The interaction between a decision maker and an environment is commonly modeled as a *Markov decision process* (MDP). A Markov decision process (MDP) (Puterman, 1994; Kaelbling et al., 1998) is a mathematical formulation of the problem of sequential decision making under uncertainty. It can be defined by a tuple $\langle S, A, \mathcal{T}, \mathcal{R} \rangle$, where

- S is the set of states of the environment,
- A is the set of actions,
- $\mathcal{T}$ is a transition (probability) function, $\mathcal{T}(s, a, s') = p(s' \mid s, a)$, and
- $\mathcal{R}$ is a reward function, $\mathcal{R}(s, a) = \mathrm{E}[r \mid s, a]$,

where p denotes the probability, E denotes the expectation, $s \in S$, $s' \in S$, $a \in A$, and $r \in \mathbb{R}$.

The interaction between the decision maker and the environment can be denoted as:

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2 \ldots, s_t, a_t, r_{t+1}, s_{t+1}, \ldots \tag{2.1}$$

The decision maker begins in an initial state s_0. At each time step t, it observes the current state of the environment $s_t \in S$ and executes an action $a_t \in A$, then it receives a reward r_{t+1} specified by the reward function $\mathcal{R}(s_t, a_t) = \mathrm{E}[r_{t+1} \mid s_t, a_t]$ and finds itself in a new state $s_{t+1} \in S$, which is specified by the transition function $\mathcal{T}(s_t, a_t, s_{t+1}) = p(s_{t+1}|s_t, a_t)$.

The modifier "Markov" is used because the decision making process satisfies the Markov assumption, which requires that the state transition probability function and reward function depend only on the current state of the environment and the decision maker's action, and not on the history of states and actions. That is,

$$p(s_{t+1}, r_{t+1}|s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0) = p(s_{t+1}, r_{t+1}|s_t, a_t) \qquad (2.2)$$

The solution to an MDP is called a policy. It is a mapping π: $S \times A \rightarrow [0, 1]$ where $\pi(s, a)$ is the probability of taking action a in state s. The quality of a policy can be measured by various optimality criteria for a given problem.

2.1 Optimality Criteria for MDPs

In an MDP there is an obvious tradeoff between obtaining immediate rewards and guiding the process toward future states that yield potentially greater rewards. Therefore, the optimality criteria or optimization objectives are usually defined as a function of the future rewards obtained. That is, the goal of a decision maker in an MDP is to find a policy that maximizes the reward obtained over the long run. However, there are different ways to define long run reward in an MDP and these can affect the choice of the optimal policy.

Undiscounted Reward for Episodic Tasks. In episodic (or finite horizon) tasks, the environment has one or more terminal states. All the transitions from the terminal state go back to itself with probability one and reward zero. The decision maker's goal is to maximize the expected sum of the reward obtained over a finite episode $t = 0, ..., T - 1$, where T may be a random variable,

$$\mathrm{E}[r_1 + r_2 + \cdots + r_T] \qquad (2.3)$$

Discounted Reward for Continuing Tasks. In continuing (or infinite horizon) tasks, the sum of the rewards can be infinite as the decision maker might take an infinite number of actions. Moreover, short-term rewards are preferred over long-term rewards. Therefore, the objective is to maximize the total expected discounted reward

$$\mathbf{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots\right] \tag{2.4}$$

where $0 \leq \gamma < 1$ and all rewards are bounded. Note that if we let $\gamma = 1$, episodic problems can be viewed as a special case of continuing problems: the discounted and undiscounted reward measures agree for any finite horizon tasks.

In this setting, it is useful to define the notion of a *value function*. The *value* for a policy π in state s_t is defined as the expected discounted sum of future rewards starting from state s_t and following policy π thereafter

$$V^{\pi}(s_t) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \middle| s_0 = s_t, \pi\right] \tag{2.5}$$

The goal of the decision maker is to follow a policy π^* that maximizes the value function in every state; that is, to follow a policy π^* such that

$$V^{\pi^*}(s) \geq V^{\pi}(s) \quad \forall \pi \; \forall s \in S \tag{2.6}$$

The Bellman equation relates the value of a state to the values of its possible successor states

$$V^{\pi}(s_t) = \sum_{a_t \in A} \pi(s_t, a_t) \left(\mathcal{R}(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} \mathcal{T}(s_t, a_t, s_{t+1}) V^{\pi}(s_{t+1}) \right) \tag{2.7}$$

It can be shown that any function that satisfies the following "optimal" form of Bellman's equation must also be the value function of the optimal policy (Bertsekas & Tsitsiklis, 1996). That is, if V^* satisfies

$$V^*(s_t) = \max_{a_t \in A} \left(\mathcal{R}(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} \mathcal{T}(s_t, a_t, s_{t+1}) V^*(s_{t+1}) \right) \tag{2.8}$$

then

$$V^*(s_t) = \max_{\pi} V^{\pi}(s_t) \quad \forall s_t \in S \tag{2.9}$$

Similarly, the *action value*, also called the Q-function for a policy π in state s_t is defined as the expected discounted sum of future rewards starting from state s_t and taking action a_t, then following policy π thereafter

$$Q^{\pi}(s_t, a_t) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \middle| s_0 = s_t, a_0 = a_t, \pi\right] \tag{2.10}$$

The Bellman action value equation is:

$$Q^{\pi}(s_t, a_t) = \mathcal{R}(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} \mathcal{T}(s_t, a_t, s_{t+1}) \sum_{a_{t+1} \in A} \pi(s_{t+1}, a_{t+1}) Q^{\pi}(s_{t+1}, a_{t+1}) \tag{2.11}$$

Similar to above, any function that satisfies the following form of Bellman's equation must also be the action value function of the optimal policy. That is, if Q^* satisfies

$$Q^*(s_t, a_t) = \mathcal{R}(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} \mathcal{T}(s_t, a_t, s_{t+1}) \max_{a_{t+1} \in A} Q^*(s_{t+1}, a_{t+1}) \tag{2.12}$$

then

$$Q^*(s_t, a_t) = \max_{\pi} Q^{\pi}(s_t, a_t) \quad \forall s_t \in S, a_t \in A \tag{2.13}$$

Solving an MDP often refers to computing the optimal state value function V^* or the optimal state-action value function Q^*, since the optimal policy can be recovered from V^* or Q^* by taking greedy action with respect to the optimal values, i.e., $\text{argmax}_a Q^*(s, a) = \text{argmax}_a \mathcal{R}(s, a) + \gamma \sum_{s'} \mathcal{T}(s, a, s') V^*(s')$. Note that the optimal policy can be directly recovered from Q^*, but requires the model ($\mathcal{T}$ and $\mathcal{R}$) to be recoverable from V^*.

Average Reward for Undiscounted Continuing Tasks. Average reward for undiscounted continuing tasks is defined as the long term reward per time step:

$$\lim_{T \to \infty} \frac{\mathbf{E}\left[r_1 + r_2 + \cdots + r_T\right]}{T} \tag{2.14}$$

Average reward does not depend on the starting state, assuming that any state can reach any other state under any policy (i.e., we assume the decision process is *ergodic*). Thus, in the long run the average reward is the same from any state (Sutton

& Barto, 1998). I have not been focusing on the average reward case in my research, so do not consider it further here.

So far, I have defined the MDP model and the optimization objectives. The next task is to solve it. Below, I organize the solution techniques for MDPs as planning algorithms, learning algorithms, and hybrid techniques.

2.2 Solution Techniques for MDPs

Solving an MDP means finding an optimal policy for it. Under general conditions, for a fully specified MDP there is always a deterministic policy π^*: $S \rightarrow A$ that gives the optimal action in each state (Bertsekas, 1995). An optimal policy can be found by directly searching the space of policies (*policy search based methods*),[1] or by computing the optimal value function first and then deriving an optimal policy from the value function by choosing greedy actions with respect to it (*value function based methods*), as shown above. Furthermore, depending whether the MDP model (state transition probability function and reward function) is known or not, value function based approaches can be categorized as *planning* algorithms that require the knowledge of an environment model, or *learning* algorithms that do not require a model.

2.2.1 Planning Algorithms

If the MDP model is known, then Equation 2.8 (or Equation 2.12) defines a system of equations whose solution is the optimal value function (action value function). These equations can be solved directly either by linear programming or via dynamic programming. These algorithms are called *planning algorithms* for MDPs because they require knowledge of the environment ($\mathcal{T}$ and $\mathcal{R}$).

Linear programming is a procedure for locating the maximum or minimum of

[1] The idea of policy gradient search algorithms is to parameterize a policy, estimate the gradient of a performance measure of the policy (e.g., average reward over time) with respect to the policy parameters, then improve the policy by moving the parameters in the direction of the gradient (Marbach & Tsitsiklis, 2000; Baxter & Bartlett, 2001; Ng et al., 1999b; Ng & Jordan, 2000; Strens & Moore, 2002; Kakade et al., 2003)

a linear function of variables that are subject to linear constraints. A form of linear program that can solve the MDP planning problem is given by Bertsekas (1995)

$$\begin{aligned} &\min_V \sum_s V(s) \quad \text{subject to} \\ &V(s) \geq \mathcal{R}(s,a) + \gamma \textstyle\sum_{s'} \mathcal{T}(s,a,s')V(s') \qquad \forall s, a \end{aligned} \tag{2.15}$$

Dynamic programming is a method of solving problems with the properties of overlapping subproblems and optimal substructure.[2] The two standard dynamic programming algorithms for solving MDPs are *policy iteration* and *value iteration* (Bertsekas, 1995).

Policy Iteration. The policy iteration algorithm (see Table 2.1) iteratively evaluates a policy, computes the value function for a given policy, and improves the policy by choosing the greedy action with respect to the current estimates of the value function. One drawback of the policy iteration algorithm is that the policy evaluation phase can be costly since it either requires a sizeable linear system to be solved or requires multiple sweeps through every state to compute the value function iteratively. However, the algorithm is guaranteed to converge to an optimal policy in a finite number of policy-update iterations (Bertsekas, 1995).

Value Iteration. The value iteration algorithm (see Table 2.2) tries to avoid the drawback of a doubly nested loop, as in the policy iteration algorithm. This algorithm uses the update form of the Bellman optimal value function (see Equation 2.8)

$$V(s) \leftarrow \max_a \left(\mathcal{R}(s,a) + \gamma \sum_{s' \in S} \mathcal{T}(s,a,s')V(s') \right) \tag{2.16}$$

to approximate the optimal value function iteratively. Once an optimal value function is found, the optimal policy can be recovered by choosing the greedy action in each state

$$\pi^*(s) = \operatorname*{argmax}_a \left(\mathcal{R}(s,a) + \gamma \sum_{s' \in S} \mathcal{T}(s,a,s')V(s') \right) \tag{2.17}$$

[2]Optimal substructure means that optimal solutions of subproblems can be used to find the optimal solutions of the overall problem.

Note that recovering the greedy policy π^* from the optimal state value function requires knowing the model ($\mathcal{T}$ and $\mathcal{R}$). The value function obtained with value iteration is only guaranteed to converge to the optimal value function in the limit. However, a value function sufficient to recover the optimal policy will be found after a finite number of iterations (Bertsekas, 1995). In my work below, I will refer to this as the Bellman Iteration algorithm, for reasons that will become clear in Chapter 3.

2.2.2 Learning Algorithms

If the model is unknown, one can still estimate the optimal value (action value) function simply by interacting with the environment. There are two standard approaches to estimating the optimal value function. The first techniques are based on *temporal difference* (TD) learning (Sutton, 1984) or Monte Carlo Methods (Michie & Chambers, 1968) (referred to as *value-based learning* or *model free learning*), which attempt to estimate the optimal value function directly without ever attempting to learn the model ($\mathcal{T}$ and $\mathcal{R}$). The second approach is to directly learn a model and then plan using it (referred to as *model-based learning*).

Most research in reinforcement learning has considered estimating the optimal action value function directly from experience. For this purpose, two of the fundamental TD methods are Q-learning (Watkins, 1989) and *Sarsa* (Sutton & Barto, 1998).

The Q-learning algorithm (see Table 2.3) updates the value of an executed action from a sample of the next state with max operator over next actions,[3] and can learn the optimal value function while following another policy (referred to as *off-policy learning*). The *Sarsa* algorithm (see Table 2.4), on the other hand, estimates the action value function for a policy by a sample of the next state,[4] and improve this policy by being greedy with respect to the estimated value, then follows the improved policy (referred to as *on-policy learning*). These algorithms both converge to optimal action value function with probability 1 as long as all the state-action

[3] That is, $\max_{a'} Q(s', a')$.
[4] That is, $Q(s', a')$.

Algorithm: Policy Iteration

1. Initialization
   For each $s \in S$
      $V(s) \leftarrow$ arbitrary value
      $\pi(s) \leftarrow$ admissible action in state s
2. Policy Evaluation
   Repeat
      $\Delta_{max} \leftarrow 0$
      For each $s \in S$
         $v(s) \leftarrow V(s)$
         $V(s) \leftarrow \mathcal{R}(s, \pi(s)) + \gamma \sum_{s' \in S} \mathcal{T}(s, \pi(s), s') V(s')$
         $\Delta_{max} \leftarrow \max\left(\Delta_{max}, |v(s) - V(s)|\right)$
   until $\Delta_{max} < \epsilon$ (a small positive number)
3. Policy Improvement
   policy-stable $\leftarrow$ true
   For each $s \in S$
      best-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \text{argmax}_a \left(\mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{T}(s, a, s') V(s')\right)$
      If *best-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ false
   If *policy-stable*, then stop; else go to 2

Table 2.1: The policy iteration algorithm

Algorithm: Value Iteration

Initialize $V(s)$ arbitrarily
Repeat
   $\Delta_{max} \leftarrow 0$
   For each $s \in S$
      $v(s) \leftarrow V(s)$
      $V(s) \leftarrow \max_a \left(\mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{T}(s, a, s') V(s')\right)$
      $\Delta_{max} \leftarrow \max\left(\Delta_{max}, |v(s) - V(s)|\right)$
until $\Delta_{max} < \epsilon$ (a small positive number)
Output a deterministic policy π
   $\pi(s) \leftarrow \text{argmax}_a \left(\mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{T}(s, a, s') V(s')\right)$

Table 2.2: The value iteration algorithm

Algorithm: Q-learning

For each $s \in S, a \in A$
  Initialize $Q(s,a)$ (**How?** e.g. arbitrarily)

Repeat (for each episode):
  Initialize s (**How?** e.g. arbitrarily)
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (**How?** e.g. ϵ-greedy)
    Take action a and observe the reward r and next state s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big)$
    $s \leftarrow s'$
  until s is terminal

Table 2.3: The Q-learning algorithm

Algorithm: Sarsa

For each $s \in S, a \in A$
  Initialize $Q(s,a)$ (**How?** e.g. arbitrarily)

Repeat (for each episode):
  Initialize s (**How?** e.g. arbitrarily)
  Choose a from s using policy derived from Q (**How?** e.g. ϵ-greedy)
  Repeat (for each step of episode):
    Take action a and observe the reward r and next state s'
    Choose a' from s' using policy derived from Q (**How?** e.g. ϵ-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$
    $s \leftarrow s'$ $a \leftarrow a'$
  until s is terminal

Table 2.4: The Sarsa algorithm

pairs are visited an infinite number of times (Watkins & Dayan, 1992; Tsitsiklis, 1994a; Singh et al., 2000).

In Table 2.3 and Table 2.4, I insert the label "**How?**" to emphasize the connection between my research in Chapter 4 and these existing popular reinforcement learning algorithms. My research in Chapter 4 contributes to offering solutions to these "**How?**" questions.

2.2.3 Hybrid Solution Techniques for MDPs

When the MDP model is not given, one may attempt to estimate the transition function and reward function by interacting with the environment first, and then determining a policy by solving the planning problem in the learned model. This is the model based approach. In addition, many hybrid methods have been proposed, such as actor-critic (Sutton, 1984), Dyna (Sutton, 1991), and prioritized sweeping (Moore & Atkeson, 1993), which combine both a model-based learning approach with a direct value-based learning approach.

Specifically, actor-critic methods are temporal difference methods that have both a policy structure for action selection (*actor*) and an estimated state value function (*critic*). Dyna-Q is another architecture for integrating planning, leaning, and acting. The idea is to learn a model from real experience, and then use the model to generate simulated experience, which can be used to update the value function. In the Dyna architecture, the value function update starts with a random state-action pair that is uniformly sampled from the simulated experience. This is interleaved with updates based on real experience, using either Sarsa or Q-leaning. Prioritized sweeping is an improvement to the experience simulator portion of Dyna, which ranks the state-action pairs according to how much their value is likely to have changed.

2.3 Approximate Solution Techniques for MDPs

In order to cope with large sequential decision making problems, many approaches have been proposed. However, I will only focus on the ones that are closely relevant

to my research. These approximate approaches can be categorized as value function approximation, sampling, and exploiting model structure.

2.3.1 Value Function Approximation

Value function approximation approach generalize across state space or state-action pair space by exploiting the structure of the problem. It is mainly applied to scale up learning algorithms to large sequential decision making problem domains. It can also be used for planning algorithms.

The simplest and most common approach to representing a value function with a compact approximation is to parameterize the value function as some simple function of features and adjust the parameters based on gradient principles. The most popular approach, for example, is to approximate the state value function $V(s)$ as a *linear* function of the parameter vector $\mathbf{w}$ (an $k \times 1$ vector) and features of the state $\Phi(s)$ (an $k \times 1$ vector); that is

$$\hat{V}(s) = \Phi(s)^\top \mathbf{w} \tag{2.18}$$

The gradient of the approximated value function with respect to $\mathbf{w}$ is then given by

$$\nabla_{\mathbf{w}} \hat{V}(s) = \Phi(s) \tag{2.19}$$

The TD update techniques mentioned above (Table 2.3 and Table 2.4) can be modified to work with linear value function approximations instead of the simple tabular representations considered before; see for example (Sutton & Barto, 1998). It is also possible to approximate value function with nonlinear function, e.g., by using a neural network.

Many successful applications of reinforcement learning with function approximation have been reported in the literature. For instance, TD(λ) with nonlinear function approximation created the famous world-level backgammon master player (Tesauro, 1994); Sarsa(λ) with linear function approximation has been applied to control the acrobot (Sutton, 1996) and to learn decisions in a keepaway task of RoboCup soccer (Stone et al., 2005); reinforcement learning algorithms were used for autonomous control of a helicopter (Ng et al., 2004; Abbeel et al., 2007).

Although there are many successful examples of reinforcement learning algorithms being applied to practical problems, there are also examples that show some forms of approximate reinforcement learning algorithms can lead to instability or divergence (Bradtke, 1993; Thrun & Schwartz, 1993; Boyan & Moore, 1995; Baird, 1995; Gordon, 2000). Therefore, it is useful to understand the convergence properties of these approximate algorithms.

In Chapter 3, I will present my research work on dual representations, dual dynamic programming algorithms, and their convergence analysis, which will address some of these convergence issues.

2.3.2 Sampling

A completely different approach to approximating value functions in large state spaces is never to keep an explicit representation, but rather directly estimate the values of visited states as they are encountered. This requires access to either interaction experience or a generative model, but not an explicit representation of the environment. Therefore, sampling is generally applied to planning.

The Monte Carlo approach is a general strategy for approximating values in Markov decision processes. It estimates value functions based on averaging over samples either from real experience or simulated experience. Since Monte Carlo methods can focus on only visited states, they are efficient in updating value estimates, and may still work even when the Markov assumption of a decision process is violated. The details of Monte Carlo algorithms can be found in (Sutton & Barto, 1998).

Different from Monte Carlo methods in averaging sample trajectories, the *sparse sampling* approach (Kearns et al., 2001) samples experience with some structure (e.g., shared states). The idea of sparse sampling is to enumerate the actions at the decision nodes and sample the rewards at the outcome nodes to generate a tree with a fixed depth (see Figure 2.1). Then, it evaluates the value of the nodes in the sparse lookahead tree from leaves to the root by alternatively computing *expectation* at the outcome nodes and *max* at the decision nodes. Finally, the optimal action at the current state (root node) is the greedy action with respect to the estimated optimal

value at the root. A generic outline of the sparse sampling algorithm for finite horizon problems is given in Table 2.5. Note that sparse sampling requires a generative model to generate the next state and reward of a given action during the lookahead procedure.

Although sparse sampling requires significant computation to determine each action value, it still has some advantages. First, as Kearns and his colleagues (2001) show, it is guaranteed to produce a near optimal action value for *any* state encountered. Second, sparse sampling can be easily applied to *infinite* state spaces. As Table 2.5 shows, this procedure can be parameterized so that the computational cost can be controlled by making the outcome branching factor and lookahead depth inputs to the procedure. This requires us to give up the theoretical guarantees of near-optimality, but that should not be surprising, since guaranteed approximation in this case is still provably intractable (Mundhenk et al., 2000; Lusena et al., 2001).

Some subsequent work has attempted to improve the practical efficiency of sparse sampling. One such example is Péret & Garcia strategy (2004), which is an on-line search strategy for action selection in MDPs. Motivated by sparse sampling, Péret and Garcia construct a lookahead tree by sampling fixed-length trajectories from the current state, employing Boltzmann selection to choose the actions along each trajectory. Then the optimal action at the root is the one with the best overall trajectory reward on average.

Below in Chapter 4, I develop an approach, Bayesian sparse sampling, that demonstrates further improvements.

2.3.3 Exploiting Model Structure

Another approach to scaling up is to directly exploit representational structure to reduce the computational cost of storing and computing value function approximations (Boutilier et al., 1999). This approach generally assumes there is a compact description of the transition model $\mathcal{T}$—either a compact decision diagram (Boutilier et al., 1999) or a dynamic Bayesian network (Koller & Parr, 1999)—and a compact description of the reward function. Unfortunately, given such a compact description of the domain, it usually does not follow that the optimal value functions also

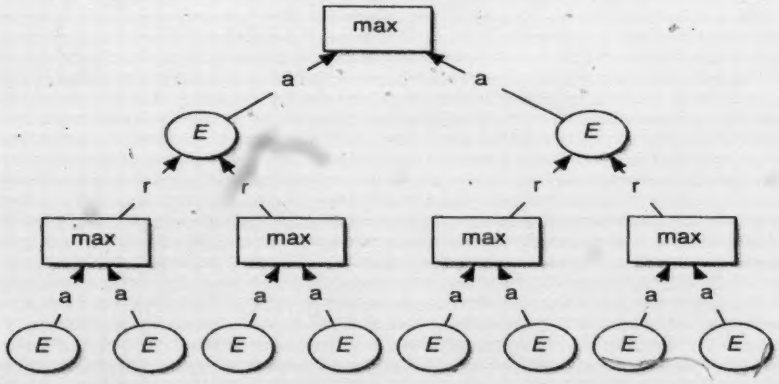

Figure 2.1: Illustration of a lookahead tree, showing decision (max) nodes and outcome (expectation) nodes. Once built, optimal value and action estimates are backed up to the root.

```
GrowSparseTree (node, branchfactor, horizon)
    If node.depth = horizon; return
    If node.type = "decision"
      For each a ∈ A
          child = ("outcome", depth, node.belstate, a)
          GrowSparseTree (child, branchfactor, horizon)
    If node.type = "outcome"
      For i = 1...branchfactor
          [rew,obs] = sample(node.belstate, node.act)
          post = posterior(node.belstate, obs)
          child = ("decision", depth+1, post, [rew,obs])
          GrowSparseTree (child, branchfactor, horizon)

EvaluateSubTree (node, horizon)
    If node.children = empty
      immed = MaxExpectedValue(node.belstate)
      return immed * (horizon - node.depth)
    If node.type = "decision"
      return max(EvaluateSubTree(node.children))
    If node.type = "outcome"
      values = EvaluateSubTree(node.children)
      return avg(node.rewards + values)
```

Table 2.5: Sketch of the sparse sampling algorithm. Grows a balanced lookahead tree, enumerating actions at decision nodes and sampling at outcome nodes. Sufficiently large values of "branchfactor" and "horizon" yield approximation guarantees.

have a compact description (Koller & Parr, 1999; Guestrin et al., 2003). Therefore, one usually has to assume a compact form of value function approximation in addition—either another decision diagram (Boutilier et al., 1999) or a linear value function approximation (Koller & Parr, 1999). Given these approximations, an efficient (although approximate) form of dynamic programming (Boutilier et al., 1999; Koller & Parr, 2000) or linear programming (Guestrin et al., 2003; Schuurmans & Patrascu, 2001) can be used to efficiently compute an approximate solution.

The main drawback of these approaches is that the algorithms are very complex and do not provide strong guarantees about the quality of their results (Patrascu et al., 2002).

2.4 Partially Observable Markov Decision Processes

Another formal specification of sequential decision making problems are *Partially Observable Markov Decision Processes* (POMDPs), which relax the assumption of full observability of the state of the environment. POMDPs are a general framework for optimal decision making under uncertainty and have a wide range of applications such as robot planning (Spaan & Vlassis, 2004) and health-care robotics (Pineau et al., 2003a). A POMDP model is defined by a tuple $\langle S, A, \mathcal{T}, \mathcal{R}, \Omega, O \rangle$, where

- S, A, $\mathcal{T}$, and $\mathcal{R}$ define the underlying Markov decision process of the POMDP,
- Ω is the set of observations,
- O is a observation function, O: $S \times A \rightarrow \Omega$.

Similar to an MDP, a POMDP is defined by a set of states, a set of actions, a state transition function, and a reward function. However, POMDPs extend MDPs by introducing an observation function $p(o'|a, s')$ that governs how a noisy observation $o' \in O$ is related to the underlying state s' and the action a. Having access to only noisy observations of the state complicates the problem of choosing optimal actions significantly. The agent generally does not know the exact state of the environment, but instead must infer a distribution over possible states, a "belief state", from the history of observations and actions. Nevertheless, a belief state in a POMDP is a

sufficient summary of history for decision making and this notion of state can be updated by Bayes' rule (see Section 4.2).

Learning a POMDP model means learning an observation function O, transition function $\mathcal{T}$, and reward function $\mathcal{R}$. Given a model (i.e, knowing these functions), finding a policy for action selection is called POMDP planning. The solution to a POMDP is a policy, which specifies the best action to take for every belief state. As one can see, the set of belief states is generally continuous and high dimensional. Computing an optimal policy for a POMDP is known to be hard and even approximate POMDP planning has proven to be a significant challenge (Mundhenk et al., 2000; Madani et al., 2003).

Therefore, some form of approximation is important here, perhaps even more so than in the simpler MDP case. Algorithms to POMDP planning can be categorized as value function approximation (Parr & Russell, 1995; Hauskrecht, 2000; Pineau et al., 2003a; Spaan & Vlassis, 2005), policy based optimization (Ng & Jordan, 2000; Poupart & Boutilier, 2002; Poupart & Boutilier, 2003; Poupart & Boutilier, 2004; Amato et al., 2006) and stochatic sampling (Thrun, 2000; Kearns et al., 2002). Specific approximation strategies relevant to my research will be described further in Chapter 5.

In Chapter 5, I will present my approach for approximating POMDP planning.

2.5 Summary

So far I have reviewed models of sequential decision making, MDPs and POMDPs, and some standard algorithms for planning and learning in these models exactly and approximately. In Chapter 3, 4, and 5, I will present my new approximate strategies for planning and learning in uncertain environments. My research is motivated by trying to address three key outstanding problems in sequential decision making under uncertainty.

- Is it possible to do off-policy updates with function approximation while avoiding the risk of divergence? If possible, how?
- How can we balance the fundamental trade-off between exploration and ex-

ploitation during action selection?

- How can we deal with partial observability in MDPs efficiently in terms of memory and computation?

Each of these questions will be addressed in turn in the next three chapters.

Chapter 3

Dual Representations and Dual Algorithms

In this chapter, I propose to use a new dual approach to dynamic programming. The idea is to maintain an explicit representation of visit distributions as opposed to value functions. A significant advantage of the dual approach is that it allows one to exploit well developed techniques for representing, approximating, and estimating probability distributions, without running the risks associated with divergent value function estimation. I introduce novel dual representations and develop dual algorithms. Moreover, I show that dual dynamic programming algorithms remain stable in situations where standard value function estimation diverges. I also show that the dual view yields a viable alternative to standard value function based techniques and opens new avenues for solving sequential decision making problems.

In some sense, reinforcement learning algorithms can be viewed as sampled versions of dynamic programming algorithms. That is, we can get the corresponding learning algorithms with point-wise updates (RL algorithms) by sampling the model of an MDP in these planning algorithms (DP algorithms). Therefore, understanding the underlying updates of dynamic programming algorithms will help understand reinforcement learning algorithms. For clarity of the presentation, dual representations, dual dynamic programming algorithms, and their convergence analysis are presented in this chapter. Dual reinforcement learning algorithms are given in Appendix B (tabular case) and Appendix C (approximate case).

This chapter is organized as follows. Section 3.1 motivates the idea. Section 3.2

gives the notation used in the later presentation. Section 3.3 presents a modified dual of the standard linear program that guarantees a globally normalized state visit distribution is obtained. Novel dual forms of policy evaluation, policy improvement, and Bellman iteration algorithms are organized according to their underlying updates in sections 3.4, 3.5, and 3.6. Section 3.7 shows how linear approximation can be used with dual representations, allowing it to be applied to large problems. Section 3.8 demonstrates that the advantage of dual dynamic programming algorithms through experimental studies on randomly synthesized MDPs, the star problem, and the mountain car problem. Section 3.9 gives the complexity analysis of both primal and dual dynamic programming algorithms. Finally, Section 3.10 summarizes the strengths and weaknesses of the dual representations.

3.1 Motivation

Algorithms for dynamic programming (DP) and reinforcement learning (RL) are usually formulated in terms of *value functions*—representations of the long run expected value of a state or state-action pair (Sutton & Barto, 1998). The concept of value is so pervasive in DP and RL, in fact, that it is hard to imagine that a value function representation is not a necessary component of any solution approach. Yet, linear programming (LP) methods clearly demonstrate that the value function is *not* a necessary concept for solving DP problems. In LP methods, value functions only correspond to the primal formulation of the problem, and do not appear at all in the dual. Rather, in the dual, value functions are replaced by the notion of state (or state-action) *visit distributions* (Puterman, 1994; Bertsekas, 1995; Bertsekas & Tsitsiklis, 1996). It is entirely possible to solve DP and RL problems in the dual representations, which offer an equivalent but different approach to solving decision-making problems without any reference to value functions.

Despite the well known LP duality, dual representations have not been widely explored in DP and RL. In fact, they have only been anecdotally and partially treated in the RL literature (Dayan, 1993; Ng et al., 1999a), and not in a manner that acknowledges any connection to LP duality. Nevertheless, as I will show,

there exists a dual form for every standard DP and RL algorithm, including policy evaluation, policy iteration, Bellman iteration, temporal difference (TD) estimation, Sarsa learning, and Q-learning, and for variants of these algorithms that use linear approximation. Reinforcement learning algorithms can be found in Appendix B (tabular case) and Appendix C (approximate case).

In this chapter, I offer a systematic investigation of dual solution techniques based on representing state visit and state-action visit distributions instead of value functions in the planning context (i.e. given an environmental model). Although many of my results show that the dual dynamic programming approach yields equivalent results to the primal approach in tabular case —as one would expect— the dual approach has potential advantage over the primal in approximate case as I will show below. The dual view offers a coherent and comprehensive perspective on optimal sequential decision making problems, just as the primal view, but offers new algorithmic insight and new opportunities for developing algorithms that exploit alternative forms of prior knowledge and constraints. In fact, there is the opportunity to develop a joint primal-dual view of sequential decision making under uncertainty, where combined algorithms might be able to exploit the benefits of both approaches in theoretically justified ways.

3.2 Preliminaries

For the easy of exposition and elegance of presentation, I will express the transition function and the reward function of an MDP, which I introduced in Chapter 2, in a matrix form below.

- The transition model is expressed as an $|S||A|\times|S|$ *transition matrix* P, whose entries $P_{(sa,s')}$ specify the conditional probability of transitioning to state s' starting from state s and taking action a (hence P is nonnegative and *row* normalized), i.e., $P_{(sa,s')} = \mathcal{T}(s,a,s') = p(s' \mid s,a)$, where $p(s' \mid s,a) \geq 0$ and $\sum_{s'} p(s' \mid s,a) = 1 \ \forall \ s,a$.

- The reward function is expressed as an $|S||A| \times 1$ *reward vector* $\mathbf{r}$, whose entries $\mathbf{r}_{(sa)}$ specify the reward obtained when taking action a in state s, i.e.,

$$\mathbf{r}_{(sa)} = \mathcal{R}(s,a) = \mathrm{E}[r \mid s,a].$$

In this chapter, I will use $\mathbf{1}$ for a vector with all ones, whose dimension will be clear from context.

Here I focus on addressing the problem of computing an optimal behavior strategy for an MDP where the optimality criterion is maximizing the infinite horizon *discounted* reward $r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots = \sum_{t=1}^{\infty} \gamma^{t-1} r_t$ given a discount factor $0 \leq \gamma < 1$. It is known that an optimal behavior strategy can always be expressed by a stationary *policy*. In this chapter, I will represent a stationary policy by an $|S||A| \times 1$ *vector* $\boldsymbol{\pi}$, whose entries $\boldsymbol{\pi}_{(sa)}$ specify the probability of taking action a in state s; that is $\sum_a \boldsymbol{\pi}_{(sa)} = 1$ for all s. Stationarity refers to the fact that the action selection probabilities do not change over time. In addition to stationarity, it is known that furthermore there always exists a *deterministic* policy that gives the optimal action in each state (i.e., simply a policy with probabilities of 0 or 1) (Bertsekas, 1995).

The main problem is to compute an optimal policy given either a complete specification of the environmental model P and $\mathbf{r}$ (the "*planning problem*"), or limited access to the environment through observed states and rewards and the ability to select actions to cause further state transitions (the "*learning problem*"). The planning problem is normally tackled by linear programming or dynamic programming methods (see Section 2.2.1), whereas the learning problem is solved by reinforcement learning methods (see Section 2.2.2).

3.3 Linear Programming

To establish the dual form of representation, I begin by briefly reviewing the LP approach for solving MDPs in the discounted reward case. Here I assume we are given the environmental model P and $\mathbf{r}$, the discount factor γ, and the initial distribution over states, expressed by an $|S| \times 1$ vector $\boldsymbol{\mu}$.

A standard LP for solving the planning problem can be expressed as

$$\begin{aligned} &\min_{\mathbf{v}} (1-\gamma)\boldsymbol{\mu}^{\top}\mathbf{v} \quad \text{subject to} \\ &\mathbf{v}_{(s)} \geq \mathbf{r}_{(sa)} + \gamma P_{(sa,:)}\mathbf{v} \qquad \forall s,a \end{aligned} \tag{3.1}$$

where $\mathbf{v}$ is the state value function (see Equation 3.9 for the matrix form definition). By introducing an $|S| \times |S||A|$ marginalization matrix, which is built by placing $|S|$ row blocks of length $|A|$ in a block diagonal fashion, where each row block consists of all 1s

$$\Xi = \begin{pmatrix} 1 \cdots 1 & & & & \\ & 1 \cdots 1 & & & \\ & & 1 \ldots 1 & & \\ & & & \ddots & \\ & & & & 1 \cdots 1 \end{pmatrix}$$

the primal LP (see Equation 3.1) can be rewritten as

$$\begin{aligned} &\min_{\mathbf{v}} (1-\gamma)\boldsymbol{\mu}^\top \mathbf{v} \quad \text{subject to} \\ &\Xi^\top \mathbf{v} \geq \mathbf{r} + \gamma P \mathbf{v} \end{aligned} \tag{3.2}$$

That is, Ξ is constructed to simply ensure that the constraint $\Xi^\top \mathbf{v} \geq \mathbf{r} + \gamma P \mathbf{v}$ given in the above matrix form LP corresponds to the system of inequalities $\mathbf{v}_{(s)} \geq \mathbf{r}_{(sa)} + \gamma P_{(sa,:)} \mathbf{v}\ \forall s, a$ in the primal LP.

It is known that the optimal solution $\mathbf{v}^*$ to this LP corresponds to the *value function* for the optimal policy (Bertsekas, 1995; Bertsekas & Tsitsiklis, 1996). In particular, given $\mathbf{v}^*$, the optimal policy can be recovered by

$$\begin{aligned} a^*(s) &= \arg\max_a \mathbf{r}_{(sa)} + \gamma P_{(sa,:)} \mathbf{v}^* \\ \boldsymbol{\pi}^*_{(sa)} &= \begin{cases} 1 & \text{if } a = a^*(s) \\ 0 & \text{if } a \neq a^*(s) \end{cases} \end{aligned} \tag{3.3}$$

Note that $\boldsymbol{\mu}$ and $(1-\gamma)$ behave as an arbitrary positive vector and positive constant in the LP above and do not affect the solution, provided $\boldsymbol{\mu} > 0$ and $\gamma < 1$ (def,). However, both play an important and non-arbitrary role in the dual LP below (as we will see) and I have chosen the objective in Equation 3.2 in a specific way to obtain the result below.

To derive the particular form of the dual LP, which will be exploited below, I first introduce a $|S||A| \times 1$ vector of Lagrange multipliers $\mathbf{d}$, and then form the Lagrangian of the standard LP (see Equation 3.2)

$$L(\mathbf{v}, \mathbf{d}) = (1-\gamma)\boldsymbol{\mu}^\top \mathbf{v} + \mathbf{d}^\top (\mathbf{r} + \gamma P \mathbf{v} - \Xi^\top \mathbf{v}),\ \mathbf{d} \geq 0$$

Next, taking the gradient of the Lagrangian with respect to $\mathbf{v}$ and setting the resulting vector to equal zero yields

$$\Xi\mathbf{d} = (1-\gamma)\boldsymbol{\mu} + \gamma P^\top \mathbf{d}$$

Substituting this constraint back into the Lagrangian eliminates the $\mathbf{v}$ variable and results in the dual LP.

$$\begin{aligned} &\max_{\mathbf{d}} \mathbf{d}^\top \mathbf{r} \quad \text{subject to} \\ &\mathbf{d} \geq 0, \;\; \Xi\mathbf{d} = (1-\gamma)\boldsymbol{\mu} + \gamma P^\top \mathbf{d} \end{aligned} \tag{3.4}$$

Interestingly, the following lemma establishes that any feasible vector in the dual LP is guaranteed to be normalized, and therefore the solution $\mathbf{d}^*$ is always a joint *probability distribution* over state-action pairs.

Lemma 1 *If* $\mathbf{d}$ *satisfies the constraint in the dual LP (see Equation 3.4), then* $\mathbf{1}^\top \mathbf{d} = 1$.

Proof: By definition of Ξ, we have

$$\mathbf{1}^\top \mathbf{d} = \mathbf{1}^\top \Xi \mathbf{d} \tag{3.5}$$

Multiplying the constraint in Equation 3.4 by $\mathbf{1}^\top$ yields

$$\mathbf{1}^\top \Xi\mathbf{d} = (1-\gamma)\mathbf{1}^\top \boldsymbol{\mu} + \gamma \mathbf{1}^\top P^\top \mathbf{d}$$

Since P is row normalized and $\boldsymbol{\mu}$ is a probability distribution,

$$\mathbf{1}^\top \Xi\mathbf{d} = (1-\gamma) + \gamma \mathbf{1}^\top \mathbf{d} \tag{3.6}$$

From Equations 3.5 and 3.6,

$$\mathbf{1}^\top \mathbf{d} = (1-\gamma) + \gamma \mathbf{1}^\top \mathbf{d} \tag{3.7}$$

Rearranging Equation 3.7 and since $0 \leq \gamma < 1$

$$\mathbf{1}^\top \mathbf{d} = 1$$

∎

By strong duality, we know that the optimal objective value of this dual LP equals the optimal objective value of the primal LP. Furthermore, given a solution to the dual $\mathbf{d}^*$, the optimal policy can be directly recovered by the much simpler transformation (Ross, 1997)

$$\pi^*_{(sa)} = \frac{\mathbf{d}^*_{(sa)}}{\sum_a \mathbf{d}^*_{(sa)}} \tag{3.8}$$

A careful examination of the dual LP shows that the joint distribution $\mathbf{d}^*$ does *not* actually correspond to the stationary state-action visit distribution induced by π^* (unless $\gamma = 1$), but it does correspond to a distribution of discounted state-action visits beginning in the initial state distribution μ.

What this dual LP formulation establishes is that the optimal policy π^* for an MDP can be recovered without any direct reference whatsoever to the *value* function. Instead, one can work in the dual, and bypass value functions entirely, while working instead with *normalized* probability distributions over state-action pairs. Although this observation seems limited to the LP approach to solving the MDP planning problem,[1] in fact, I will show that explicit representations of probability distributions over state and state-action pairs can be used as a dual alternative to classical DP methods (see Sections 3.4, 3.5, and 3.6), classical RL methods (see Appendix B), and even classical approximation methods (see Section 3.7 and Appendix C).

Before I present dual algorithms and their convergence analysis. I find it convenient to express a policy π by an equivalent representation as an $|S| \times |S||A|$ matrix Π

$$\Pi_{(s,s'a)} = \begin{cases} \pi_{(sa)} & \text{if } s' = s \\ 0 & \text{if } s' \neq s \end{cases}$$

$$\Pi = \begin{pmatrix} p(a|s_1) & & & & \\ & p(a|s_2) & & & \\ & & p(a|s_3) & & \\ & & & \ddots & \\ & & & & p(a|s_{|S|}) \end{pmatrix}$$

[1]The dual LP formulation is equivalent to the primal LP formulation in the tabular case (i.e., no duality gap), while the approximate dual LP is different from the approximate primal LP.

where $p(a|s_1) = [\pi_{(s_1a_1)}\pi_{(s_1a_2)} \cdots \pi_{(s_1a_{|A|})}]$. That is, Π is a sparse matrix built by placing $|S|$ row blocks of length $|A|$ in a block diagonal fashion, where each row block gives the conditional distribution over actions specified by π in a particular state s. Although this representation of Π might appear unnatural,[2] I find it in fact extremely convenient in my research: from this definition, one can quickly verify that the $|S| \times |S|$ matrix product ΠP gives the *state to state* transition probabilities induced by the policy π in the environment P, and the $|S||A| \times |S||A|$ matrix product $P\Pi$ gives the *state-action to state-action* transition probabilities induced by policy π in the environment P. I will make repeated use of these two matrix products below.

Here the dynamic programming algorithms are organized according to their update types: on-policy update, policy improvement, and off-policy update. In addition both state based and state-action based representations are introduced.

3.4 On-Policy Update

First consider the problem of policy evaluation. Here I assume we are given a fixed policy π, and wish to compute either its value function in the primal or its distribution of discounted visitation distribution in the dual.

3.4.1 State Based Policy Evaluation

First let us examine the standard state based policy evaluation.

Primal Representation

In the primal view, the role of policy evaluation is to recover the *value function*, which is defined to be the expected sum of future discounted rewards (see Equation 2.5). I can now express this definition in vector-matrix form

$$\mathbf{v} = \sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \Pi \mathbf{r} \tag{3.9}$$

As is well known and easy to verify, this infinite series satisfies a recursive relationship (see Equation 3.10) that allows one to recover $\mathbf{v}$ by solving a linear system of

[2] The same definition also used in paper (Lagoudakis & Parr, 2003).

$|S|$ equations on $|S|$ unknowns.

$$\begin{aligned}
\mathbf{v} &= \sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \Pi \mathbf{r} \\
&= \Pi \mathbf{r} + \sum_{i=1}^{\infty} \gamma^i (\Pi P)^i \Pi \mathbf{r} \\
&= \Pi \mathbf{r} + \gamma (\Pi P) \sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \Pi \mathbf{r} \\
&= \Pi \mathbf{r} + \gamma \Pi P \mathbf{v} \qquad (3.10)
\end{aligned}$$

For a given policy Π, the on-policy operator $\mathcal{O}$ is defined as

$$\mathcal{O}\mathbf{v} = \Pi(\mathbf{r} + \gamma P \mathbf{v}) \qquad (3.11)$$

It brings the current representation closer to satisfying the policy-specific Bellman equation (see Equation 3.10). Therefore, we can have a solution to Equation 3.10 by iterating the on-policy operator $\mathcal{O}$. I present an analysis of convergence properties of the on-policy operator in Section 3.4.3 below.

Dual Representation

In the dual form of policy evaluation, one needs to recover a probability distribution over states that has a meaningful correspondence to the long run discounted reward achieved by the policy. Such a correspondence can be achieved by recovering the following probability distribution over states implicitly defined as

$$\mathbf{c}^\top = (1-\gamma)\boldsymbol{\mu}^\top \sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \qquad (3.12)$$

This infinite series satisfies a recursive relationship (see Equation 3.13) that allows one to recover $\mathbf{c}$ by solving a linear system of $|S|$ equations on $|S|$ unknowns.

$$\begin{aligned}
\mathbf{c}^\top &= (1-\gamma)\boldsymbol{\mu}^\top \sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \\
&= (1-\gamma)\boldsymbol{\mu}^\top + (1-\gamma)\boldsymbol{\mu}^\top \sum_{i=1}^{\infty} \gamma^i (\Pi P)^i \\
&= (1-\gamma)\boldsymbol{\mu}^\top + \gamma(1-\gamma)\boldsymbol{\mu}^\top \sum_{i=0}^{\infty} \gamma^i (\Pi P)^i (\Pi P) \\
&= (1-\gamma)\boldsymbol{\mu}^\top + \gamma \mathbf{c}^\top \Pi P \qquad (3.13)
\end{aligned}$$

It can be easily verified that Equation 3.12 defines a probability distribution.

Lemma 2 *If* $\mathbf{c}$ *satisfies the above definition (see Equation 3.12) then* $\mathbf{c}^\top \mathbf{1} = 1$.

Proof: We know that the definition of $\mathbf{c}$ satisfies Equation 3.13, then we have

$$\mathbf{c}^\top \mathbf{1} = (1-\gamma)\boldsymbol{\mu}^\top \mathbf{1} + \gamma \mathbf{c}^\top \Pi P \mathbf{1}$$

Since ΠP is row normalized and $\boldsymbol{\mu}$ is a probability distribution,

$$\mathbf{c}^\top \mathbf{1} = (1-\gamma)1 + \gamma \mathbf{c}^\top \mathbf{1}$$

Rearranging the above equation and since $0 \leq \gamma < 1$

$$\mathbf{c}^\top \mathbf{1} = 1$$

∎

Not only is $\mathbf{c}$ a proper probability distribution over states, it also allows one to easily compute the expected discounted return of the policy $\boldsymbol{\pi}$ (see Lemma 3).

Lemma 3 $(1-\gamma)\boldsymbol{\mu}^\top \mathbf{v} = \mathbf{c}^\top \Pi \mathbf{r}$

Proof: Plugging the definition of $\mathbf{v}$ (see Equation 3.9) into the left of the above lemma yields

$$(1-\gamma)\boldsymbol{\mu}^\top \mathbf{v} = (1-\gamma)\boldsymbol{\mu}^\top \sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \Pi \mathbf{r}$$

Plugging the definition of $\mathbf{c}$ (see Equation 3.12) into the right of the above lemma yields

$$\mathbf{c}^\top \Pi \mathbf{r} = (1-\gamma)\boldsymbol{\mu}^\top \sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \Pi \mathbf{r}$$

Thus,

$$(1-\gamma)\boldsymbol{\mu}^\top \mathbf{v} = \mathbf{c}^\top \Pi \mathbf{r}$$

∎

Thus, a dual form of policy evaluation can be conducted by recovering $\mathbf{c}$ from Equation 3.13. The expected discounted reward obtained by policy π starting in the initial state distribution μ can then be computed by $\mathbf{c}^\top \Pi \mathbf{r}/(1-\gamma)$ according to Lemma 3. In principle, this gives a valid form of policy evaluation in a dual representation. However, below I will find that merely recovering the state distribution $\mathbf{c}$ is inadequate for *policy improvement* (see Section 3.5), since there is no apparent way to improve π given access to $\mathbf{c}$. Thus, I am compelled to extend the dual representation to a richer representation that avoids an implicit dependence on the initial distribution μ.

Consider the following definition for an $|S| \times |S|$ matrix

$$M = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \tag{3.14}$$

This infinite series satisfies a recursive relationship that allows one to recover M by solving a linear system of $|S|$ equations on $|S|$ unknowns.

$$\begin{aligned}
M &= (1-\gamma)\sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \\
&= (1-\gamma)I + (1-\gamma)\sum_{i=1}^{\infty} \gamma^i (\Pi P)^i \\
&= (1-\gamma)I + (1-\gamma)(\gamma \Pi P)\sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \\
&= (1-\gamma)I + (\gamma \Pi P)(1-\gamma)\sum_{i=0}^{\infty} \gamma^i (\Pi P)^i \\
&= (1-\gamma)I + \gamma \Pi P M
\end{aligned} \tag{3.15}$$

Based on the above recursive relationship, each row of M can be written as

$$M = \begin{pmatrix} m_1{}^\top \\ m_2{}^\top \\ m_3{}^\top \\ \vdots \\ m_{|S|}{}^\top \end{pmatrix} = \begin{pmatrix} (1-\gamma)\mathbf{e}_1{}^\top + \gamma \Pi P m_1{}^\top \\ (1-\gamma)\mathbf{e}_2{}^\top + \gamma \Pi P m_2{}^\top \\ (1-\gamma)\mathbf{e}_3{}^\top + \gamma \Pi P m_3{}^\top \\ \vdots \\ (1-\gamma)\mathbf{e}_{|S|}{}^\top + \gamma \Pi P m_{|S|}{}^\top \end{pmatrix}$$

where $\mathbf{e}_s$ ($s = 1, \ldots, |S|$) is a vector of all zeros except for a 1 in the s^{th} position. The matrix M that satisfies this linear relation is similar to $\mathbf{c}^\top$, in that each row is a

probability distribution (Lemma 4 below) and the entries $M_{(s,s')}$ correspond to the probability of discounted state visits to s' for a policy π starting in state s. However, unlike $\mathbf{c}^\top$, M drops the dependence on $\boldsymbol{\mu}$ and obtains a close relationship with $\mathbf{v}$ (Theorem 1 below).

Lemma 4 $M\mathbf{1} = \mathbf{1}$

Proof: Plugging the definition of M (see Equation 3.14) into the left of the above lemma yields

$$M\mathbf{1} \;=\; (1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\Pi P)^i\mathbf{1}$$

Since ΠP is row normalized, we have

$$(\Pi P)^i\mathbf{1} \;=\; \mathbf{1}$$

Thus,

$$M\mathbf{1} \;=\; (1-\gamma)\sum_{i=0}^{\infty}\gamma^i\mathbf{1} \;=\; \mathbf{1}$$

∎

Lemma 5 $\mathbf{c}^\top \;=\; \boldsymbol{\mu}^\top M$

Proof: Note that the definition of $\mathbf{c}$ (see Equation 3.12) is

$$\mathbf{c}^\top \;=\; (1-\gamma)\boldsymbol{\mu}^\top\sum_{i=0}^{\infty}\gamma^i(\Pi P)^i$$

Plugging the definition of M (see Equation 3.14) into the right of the above lemma yields

$$\boldsymbol{\mu}^\top M \;=\; \boldsymbol{\mu}^\top(1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\Pi P)^i$$

Thus,

$$\mathbf{c}^\top \;=\; \boldsymbol{\mu}^\top M \tag{3.16}$$

∎

Interestingly, Lemmas 4 and 5 show that M is a variant of Dayan's "successor representation" proposed in (Dayan, 1993), but here extended to the infinite horizon

discounted case. Moreover, not only is M a matrix of probability distributions over states, it allows one to easily recover the state values of the policy π.

Similarly, the on-policy operator for a given policy Π is

$$\mathcal{O}M = (1-\gamma)I + \gamma M\Pi P \tag{3.17}$$

It brings the current representation closer to satisfying the policy-specific Bellman equation (see Equation 3.15). Therefore, we can have a solution to Equation 3.15 by iterating the on-policy operator $\mathcal{O}$ (see Section 3.4.3 for convergence results).

Relationship between Primal and Dual Representations

There exists a further connection between state value function $\mathbf{v}$ and state visit distribution M.

Theorem 1 $(1-\gamma)\mathbf{v} = M\Pi\mathbf{r}$

Proof: Plugging the definition of $\mathbf{v}$ (see Equation 3.9) into the left of the above theorem yields

$$(1-\gamma)\mathbf{v} = (1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\Pi P)^i\Pi\mathbf{r}$$

Plugging the definition of M (see Equation 3.14) into the right of the above theorem yields

$$M\Pi\mathbf{r} = \left[(1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\Pi P)^i\right]\Pi\mathbf{r}$$

Thus,

$$(1-\gamma)\mathbf{v} = M\Pi\mathbf{r}$$

■

A dual form of policy evaluation can be conducted by recovering M from Equation 3.15. Then at any time, an equivalent representation to $\mathbf{v}$ can be recovered by $M\Pi\mathbf{r}/(1-\gamma)$, as shown in the above theorem.

3.4.2 State-Action Based Policy Evaluation

Although state based policy evaluation methods like those outlined above are adequate for assessing a given policy, and eventually for formulating DP algorithms, for the RL algorithms below I will generally need to maintain joint *state-action* based evaluations.

Primal Representation

In the primal representation, the state-action value function (see Equation 2.11 for the definition) can be expressed as an $|S||A| \times 1$ vector

$$\mathbf{q} = \sum_{i=0}^{\infty} \gamma^i (P\Pi)^i \mathbf{r} \tag{3.18}$$

This state-action value function is closely related to the previous state value function (see Equation 3.10) and satisfies a similar recursive relation.

$$\mathbf{q} = \mathbf{r} + \gamma P\Pi\mathbf{q} \tag{3.19}$$

For a given policy Π, the on-policy operator is defined as

$$\mathcal{O}\mathbf{q} = \mathbf{r} + \gamma P\Pi\mathbf{q} \tag{3.20}$$

Iterating the on-policy operator $\mathcal{O}$ gives a solution to Equation 3.19, which I show formally in Section 3.4.3.

Dual Representation

To develop a dual form of state-action policy evaluation, I use the dual LP representation introduced in Section 3.3, and represent a probability distribution over state-action pairs that has a useful correspondence to the long run expected discounted rewards achieved by the policy

$$\mathbf{d}^\top = (1-\gamma)\boldsymbol{\nu}^\top \sum_{i=0}^{\infty} \gamma^i (P\Pi)^i \tag{3.21}$$

This state-action visit distribution is closed related to the previous state visit distribution (see Equation 3.13) and satisfies a similar recursive relation

$$\mathbf{d}^\top = (1-\gamma)\boldsymbol{\nu}^\top + \gamma \mathbf{d}^\top P\Pi \tag{3.22}$$

where $\boldsymbol{\nu}$ is the initial distribution over the state-action pairs,[3] whose dimension is $|S||A| \times 1$. It can be verified that this defines a probability distribution.

Lemma 6 *If* $\mathbf{d}$ *satisfies the above relation (see Equation 3.22) then* $\mathbf{d}^\top \mathbf{1} = 1$.

Proof: Since ΠP is row normalized and $\boldsymbol{\nu}$ is a probability distribution,

$$\begin{aligned} \mathbf{d}^\top \mathbf{1} &= (1-\gamma)\boldsymbol{\nu}^\top \mathbf{1} + \gamma \mathbf{d}^\top P \Pi \mathbf{1} \\ &= (1-\gamma)1 + \gamma \mathbf{d}^\top \mathbf{1} \end{aligned}$$

Rearranging the above equation and since $0 \leq \gamma < 1$

$$\mathbf{d}^\top \mathbf{1} = 1$$

∎

Not only is $\mathbf{d}$ a proper probability distribution over state-action pairs, it also allows one to easily compute the expected discounted return of the policy $\boldsymbol{\pi}$.

Lemma 7 $(1-\gamma)\boldsymbol{\nu}^\top \mathbf{q} = \mathbf{d}^\top \mathbf{r}$

Proof: Plugging the definition of $\mathbf{q}$ (see Equation 3.18) into the left of the above lemma yields

$$(1-\gamma)\boldsymbol{\nu}^\top \mathbf{q} = (1-\gamma)\boldsymbol{\nu}^\top \sum_{i=0}^{\infty} \gamma^i (P\Pi)^i \mathbf{r}$$

Plugging the definition of $\mathbf{d}$ (see Equation 3.21) into the right of the above lemma yields

$$\mathbf{d}^\top \mathbf{r} = (1-\gamma)\boldsymbol{\nu}^\top \sum_{i=0}^{\infty} \gamma^i (P\Pi)^i \mathbf{r}$$

Thus,

$$(1-\gamma)\boldsymbol{\nu}^\top \mathbf{q} = \mathbf{d}^\top \mathbf{r} \tag{3.23}$$

∎

[3] We can have the relationship $\boldsymbol{\nu}^\top = \boldsymbol{\mu}^\top \Pi$ between initial state visit distribution $\boldsymbol{\mu}$ and state-action visit distribution $\boldsymbol{\nu}$ by their definitions.

A dual form of state-action policy evaluation can be conducted by recovering $\mathbf{d}$ from Equation 3.22 and computing the expected discounted reward obtained by policy π starting in the initial state-action distribution ν by $\mathbf{d}^\top \mathbf{r}/(1-\gamma)$ (Lemma 7). However, once again we will find that merely recovering the state-action distribution $\mathbf{d}$ is inadequate for *policy improvement* (see Section 3.5), since there is no apparent way to improve π given access to $\mathbf{d}$. Thus, again, I extend the dual representation to a richer representation that avoids an implicit dependence on the initial distribution ν.

Consider the following definition for an $|S||A| \times |S||A|$ matrix

$$H = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i (P\Pi)^i \tag{3.24}$$

This infinite series satisfies a recursive relationship that allows one to recover H by solving a linear system of $|S||A|$ equations on $|S||A|$ unknowns.

$$H = (1-\gamma)I + \gamma P\Pi H \tag{3.25}$$

The matrix H that satisfies this linear relation is similar to $\mathbf{d}^\top$, in that each row is a probability distribution (Lemma 8 below) and the entries $H_{(sa,s'a')}$ correspond to the probability of discounted state-action visits to $(s'a')$ for a policy π starting in state-action pair (sa). However, H drops the dependence on μ and obtains a close relationship with $\mathbf{q}$ (Theorem 2 below).

Lemma 8 $H\mathbf{1} = \mathbf{1}$

Proof: Plugging the definition of H (see Equation 3.24) into the left of the above lemma yields

$$H\mathbf{1} = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i (P\Pi)^i \mathbf{1}$$

Since $P\Pi$ is row normalized, we have

$$(P\Pi)^i \mathbf{1} = \mathbf{1}$$

Thus,

$$H\mathbf{1} = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i \mathbf{1} = \mathbf{1}$$

∎

Lemma 9 $\mathbf{d}^\top = \boldsymbol{\nu}^\top H$

Proof: Note that the definition of $\mathbf{d}$ (see Equation 3.21) is

$$\mathbf{d}^\top = (1-\gamma)\boldsymbol{\nu}^\top \sum_{i=0}^{\infty} \gamma^i (P\Pi)^i$$

Plugging the definition of H (see Equation 3.24) into the right of the above lemma yields

$$\boldsymbol{\nu}^\top H = \boldsymbol{\nu}^\top (1-\gamma) \sum_{i=0}^{\infty} \gamma^i (P\Pi)^i$$

Thus,

$$\mathbf{d}^\top = \boldsymbol{\nu}^\top H \tag{3.26}$$

∎

Not only is H a matrix of probability distributions over state-action pairs, it also allows one to easily recover the state-action values of the policy π.

For a given policy Π, the on-policy operator $\mathcal{O}$ is defined as

$$\mathcal{O}H = (1-\gamma)I + \gamma P\Pi H \tag{3.27}$$

It brings the current representation closer to satisfying the policy-specific Bellman equation (see Equation 3.25). Iterating the on-policy operator $\mathcal{O}$ gives a solution to Equation 3.25, which I show formally in Section 3.4.3.

Relationship between Primal and Dual Representations

We observe the connection between state-action value function $\mathbf{q}$ and state-action visit distribution H as follows.

Theorem 2 $(1-\gamma)\mathbf{q} = H\mathbf{r}$

Proof: Plugging the definition of $\mathbf{q}$ (see Equation 3.18) into the left of the above theorem yields

$$(1-\gamma)\mathbf{q} = (1-\gamma) \sum_{i=0}^{\infty} \gamma^i (P\Pi)^i \mathbf{r}$$

Plugging the definition of H (see Equation 3.24) into the right of the above theorem yields

$$H\mathbf{r} = \left[(1-\gamma)\sum_{i=0}^{\infty}\gamma^i(P\Pi)^i\right]\mathbf{r}$$

Thus,

$$(1-\gamma)\mathbf{q} = H\mathbf{r}$$

■

A dual form of state-action policy evaluation can be conducted by recovering H from its Bellman equation (see Equation 3.25). Then at any time, an equivalent representation to $\mathbf{q}$ can be recovered by $H\mathbf{r}/(1-\gamma)$. However, there is a many to one relationship between the dual and primal representations because the number of variables in H ($|S||A|\times|S||A|$) is more than the number of the constraints given by their relation, as shown in Theorem 2.

Relationship between State and State-action Based Representations. Finally, one can relate the state and state-action matrix representations defined above to each other. We can have the relationship between state value function $\mathbf{v}$ and state-action value function $\mathbf{q}$ in the primal representation as follows.

Lemma 10 $\mathbf{v} = \Pi\mathbf{q}$

Proof: By the definitions of $\mathbf{v}$ (see Equation 3.9) and $\mathbf{q}$ (see Equation 3.18)

$$\begin{aligned}\Pi\mathbf{q} &= \Pi\sum_{i=0}^{\infty}\gamma^i(P\Pi)^i\mathbf{r}\\ &= \sum_{i=0}^{\infty}\gamma^i(\Pi P)^i\Pi\mathbf{r} = \mathbf{v}\end{aligned}$$

■

The relationship between state visit distribution M and state-action visit distributions H in the dual representation is as follows.

Lemma 11 $M\Pi = \Pi H$

Proof: By the definitions of M (see Equation 3.14) and H (see Equation 3.24)

$$\begin{aligned} M\Pi &= (1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\Pi P)^i\Pi \\ &= (1-\gamma)\sum_{i=0}^{\infty}\gamma^i\Pi(\Pi P)^i \\ &= \Pi(1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\Pi P)^i \;=\; \Pi H \end{aligned}$$

∎

Thus, to this point, I have developed new dual representations that can form the basis for state based and state-action based policy evaluation, respectively. These are defined in terms of state distributions and state-action distributions, and do not require value functions to be computed. In the next section, I will examine the convergence properties of on-policy update with the dual representations.

3.4.3 Convergence Analysis

I first investigate whether on-policy update with the dual representations exhibit the same (or better) convergence properties to their primal counterparts. These questions will be answered in the affirmative, largely showing equivalence to the standard primal cases. The real advantage of the dual approach will arise below when function approximation is considered in Section 3.7. To keep the presentation simple, I will concentrate only on state-action based representations, $\mathbf{q}$ and H, respectively. Analogous results are easily obtained for the state based representations, $\mathbf{v}$ and M.

For the on-policy operator $\mathcal{O}$, convergence to the Bellman fixed point is easily proved in the primal case, by establishing a contraction property of $\mathcal{O}$ with respect to a specific norm on $\mathbf{q}$ vectors. Although these results are already well known, I repeat some brief details that will be helpful later.

First, to establish contraction, define a weighted 2-norm with weights given by the stationary distribution determined by the policy Π with respect to the transition model P. Let $\mathbf{z} \geq 0$ be a vector such that $\mathbf{z}^\top P\Pi = \mathbf{z}^\top$; that is, $\mathbf{z}$ is the stationary state-action visit distribution for $P\Pi$. (Note that $\mathbf{z}$ is not the same as the initial

distribution ν nor the discounted stationary distribution $\mathbf{d}$.) Let $Z = \text{diag}(\mathbf{z})$. Then define the norm

$$\|\mathbf{q}\|_{\mathbf{z}}^2 = \mathbf{q}^\top Z \mathbf{q} = \sum_{(sa)} \mathbf{z}_{(sa)} \mathbf{q}_{(sa)}^2 \tag{3.28}$$

Crucially, for this norm, a state-action transition is not an expansion.

Lemma 12 $\|P\Pi\mathbf{q}\|_{\mathbf{z}} \leq \|\mathbf{q}\|_{\mathbf{z}}$ *(Tsitsiklis & Van Roy, 1997)*

Proof: The result follows from Jensen's inequality

$$\begin{aligned}
\|P\Pi\mathbf{q}\|_{\mathbf{z}}^2 &= \sum_{(sa)} \mathbf{z}_{(sa)} \left(\sum_{(s'a')} [P\Pi]_{(sa,s'a')} \mathbf{q}_{(s'a')} \right)^2 \\
&\leq \sum_{(sa)} \mathbf{z}_{(sa)} \sum_{(s'a')} [P\Pi]_{(sa,s'a')} \mathbf{q}_{(s'a')}^2 \\
&= \sum_{(s'a')} \mathbf{q}_{(s'a')}^2 \sum_{(sa)} [P\Pi]_{(sa,s'a')} \mathbf{z}_{(sa)} \\
&= \sum_{(s'a')} \mathbf{q}_{(s'a')}^2 \mathbf{z}_{(s'a')} = \|\mathbf{q}\|_{\mathbf{z}}^2
\end{aligned}$$

■

This allows one to easily recover the fact that $\mathcal{O}$ is in fact a contraction with respect to $\|\cdot\|_{\mathbf{z}}$ in the primal case.

Lemma 13 $\|\mathcal{O}\mathbf{q}_1 - \mathcal{O}\mathbf{q}_2\|_{\mathbf{z}} \leq \gamma \|\mathbf{q}_1 - \mathbf{q}_2\|_{\mathbf{z}}$ *(Tsitsiklis & Van Roy, 1997)*

Proof: By the definition of on-policy operator $\mathcal{O}$ (see Equation 3.20), we have

$$\begin{aligned}
\|\mathcal{O}\mathbf{q}_1 - \mathcal{O}\mathbf{q}_2\|_{\mathbf{z}} &= \|\mathbf{r} + \gamma P\Pi\mathbf{q}_1 - \mathbf{r} - \gamma P\Pi\mathbf{q}_2\|_{\mathbf{z}} \\
&= \gamma \|P\Pi(\mathbf{q}_1 - \mathbf{q}_2)\|_{\mathbf{z}}
\end{aligned}$$

Together with Lemma 12, we obtain

$$\|\mathcal{O}\mathbf{q}_1 - \mathcal{O}\mathbf{q}_2\|_{\mathbf{z}} \leq \gamma \|\mathbf{q}_1 - \mathbf{q}_2\|_{\mathbf{z}}$$

■

By the contraction map fixed point theorem (Bertsekas, 1995) there exists a unique fixed point of $\mathcal{O}$ in the space of vectors $\mathbf{q}$. Therefore, repeated applications

of the on-policy operator converge to a vector $\mathbf{q}_\Pi$ such that $\mathbf{q}_\Pi = \mathcal{O}\mathbf{q}_\Pi$; that is, $\mathbf{q}_\Pi$ satisfies the policy based Bellman equation (see Equation 3.19).

For the dual representation H, the convergence of the on-policy operator can be established in a similar fashion, by first defining an appropriate weighted norm over matrices and then verifying that $\mathcal{O}$ is a contraction with respect to this norm. Define

$$\|H\|_{\mathbf{z},\mathbf{r}}{}^2 = \|H\mathbf{r}\|_{\mathbf{z}}{}^2 = \sum_{(sa)} \mathbf{z}_{(sa)} \left(\sum_{(s'a')} H_{(sa,s'a')} \mathbf{r}_{(s'a')} \right)^2 \tag{3.29}$$

It is easily verified that this definition satisfies the property of a pseudo-norm, and in particular, satisfies the triangle inequality. This weighted 2-norm is defined with respect to the stationary distribution $\mathbf{z}$, but also the reward vector $\mathbf{r}$. Thus, the magnitude of a row normalized matrix is determined by the magnitude of the weighted reward expectations it induces.

Interestingly, this definition allows us to establish the same non-expansion and contraction results as the primal case. For example, state-action transitions remain a non-expansion.

Lemma 14 $\|P\Pi H\|_{\mathbf{z},\mathbf{r}} \leq \|H\|_{\mathbf{z},\mathbf{r}}$

Proof:

$$\begin{aligned} \|P\Pi H\|_{\mathbf{z},\mathbf{r}} &= \|P\Pi(H\mathbf{r})\|_{\mathbf{z}} \\ &\leq \|H\mathbf{r}\|_{\mathbf{z}} \text{ by Lemma 12} \\ &= \|H\|_{\mathbf{z},\mathbf{r}} \text{ by Equation 3.29} \end{aligned}$$

∎

Moreover, the on-policy operator is a contraction with respect to $\|\cdot\|_{\mathbf{z},\mathbf{r}}$.

Lemma 15 $\|\mathcal{O}H_1 - \mathcal{O}H_2\|_{\mathbf{z},\mathbf{r}} \leq \gamma \|H_1 - H_2\|_{\mathbf{z},\mathbf{r}}$

Proof: By the definition of on-policy operator $\mathcal{O}$ (see Equation 3.27), we have

$$\begin{aligned} \|\mathcal{O}H_1 - \mathcal{O}H_2\|_{\mathbf{z},\mathbf{r}} &= \|(1-\gamma)I + \gamma P\Pi H_1 - (1-\gamma)I - \gamma P\Pi H_2\|_{\mathbf{z},\mathbf{r}} \\ &= \|\gamma P\Pi H_1 - \gamma P\Pi H_2\|_{\mathbf{z},\mathbf{r}} \\ &= \gamma \|P\Pi(H_1 - H_2)\|_{\mathbf{z},\mathbf{r}} \end{aligned}$$

Together with Lemma 14, we obtain

$$\|\mathcal{O}H_1 - \mathcal{O}H_2\|_{\mathbf{z},\mathbf{r}} \leq \gamma\|H_1 - H_2\|_{\mathbf{z},\mathbf{r}}$$

∎

Thus, once again by the contraction map fixed point theorem, there exists a fixed point of $\mathcal{O}$ among row normalized matrices H. Therefore, repeated applications of $\mathcal{O}$ converge to a matrix H_Π such that $H_\Pi = \mathcal{O}H_\Pi$; that is, H_Π satisfies the policy based Bellman equation for dual representations (see Equation 3.25). One subtlety here is that the dual fixed point is not unique because there is a many to one relationship between the dual and primal representations, given by $H\mathbf{r} = (1-\gamma)\mathbf{q}$ in Theorem 2.

So far I have shown that on-policy dynamic programming converges in the dual representation, without making direct reference to the primal representation. By sampling the model of an MDP (P and $\mathbf{r}$) in these planning algorithms, we can get their corresponding learning algorithms with point-wise updates (see Appendix B).

3.5 Policy Improvement

The next step is to consider mechanisms for policy improvement, which combined with policy evaluation form policy iteration algorithms capable of solving MDP planning problems.

Primal Representation

The standard primal policy improvement update is well known. Given a current policy $\boldsymbol{\pi}$, whose state value function $\mathbf{v}$ or state-action value function $\mathbf{q}$ have already been determined, one can derive an improved policy $\boldsymbol{\pi}'$ via the update

$$a^*(s) = \arg\max_a \mathbf{q}_{(sa)} \tag{3.30}$$

$$= \arg\max_a \mathbf{r}_{(sa)} + \gamma P_{(sa,:)}\mathbf{v} \tag{3.31}$$

$$\boldsymbol{\pi}'_{(sa)} = \begin{cases} 1 & \text{if } a = a^*(s) \\ 0 & \text{if } a \neq a^*(s) \end{cases} \tag{3.32}$$

The subsequent "policy improvement theorem" verifies that this update leads to an improved policy.

Theorem 3 $\Pi\mathbf{q} \leq \Pi'\mathbf{q}$ *implies* $\mathbf{v} \leq \mathbf{v}'$

Proof: Since $\mathbf{v} = \Pi\mathbf{q}$ (see Lemma 10) and $\mathbf{q} = \mathbf{r} + \gamma P\Pi\mathbf{q}$ (see Equation 3.19)

$$\mathbf{q} = \mathbf{r} + \gamma P\mathbf{v} \tag{3.33}$$

Plugging the above relation into the assumption $\Pi\mathbf{q} \leq \Pi'\mathbf{q}$,

$$\begin{aligned} \Pi(\mathbf{r} + \gamma P\mathbf{v}) &\leq \Pi'(\mathbf{r} + \gamma P\mathbf{v}) \\ \Longrightarrow \Pi\mathbf{r} + \gamma\Pi P\mathbf{v} &\leq \Pi'\mathbf{r} + \gamma\Pi' P\mathbf{v} \end{aligned} \tag{3.34}$$

Using $\mathbf{v} = \Pi\mathbf{r} + \gamma\Pi P\mathbf{v}$ (see Equation 3.10) recursively, the above inequality relation becomes

$$\begin{aligned} \mathbf{v} &\leq \Pi'\mathbf{r} + \gamma\Pi' P\mathbf{v} \\ &= \Pi'\mathbf{r} + \gamma\Pi' P\Pi(\mathbf{r} + \gamma P\mathbf{v}) \\ &\leq \Pi'\mathbf{r} + \gamma\Pi' P\Pi'(\mathbf{r} + \gamma P\mathbf{v}) \\ &= \Pi'\mathbf{r} + \gamma\Pi' P\Pi'\mathbf{r} + \gamma^2(\Pi' P)^2\mathbf{v} \\ &\vdots \\ &= \sum_{i=0}^{\infty} \gamma^i(\Pi' P)^i\Pi'\mathbf{r} \end{aligned}$$

From the definition of $\mathbf{v}$ (see Equation 3.9), we have

$$\sum_{i=0}^{\infty} \gamma^i(\Pi' P)^i\Pi'\mathbf{r} = \mathbf{v}' \tag{3.35}$$

Therefore,

$$\mathbf{v} \leq \mathbf{v}' \tag{3.36}$$

■

Dual Representation

The above development can be parallelled in the dual by first defining an analogous policy update and proving an analogous policy improvement theorem. Given

a current policy π, in the dual one can derive an improved policy π' by the update

$$a^*(s) = \arg\max_a H_{(sa,:)}\mathbf{r} \tag{3.37}$$

$$= \arg\max_a (1-\gamma)\mathbf{r}_{(sa)} + \gamma P_{(sa,:)} M\Pi\mathbf{r} \tag{3.38}$$

$$\pi'_{(sa)} = \begin{cases} 1 & \text{if } a = a^*(s) \\ 0 & \text{if } a \neq a^*(s) \end{cases} \tag{3.39}$$

In fact, by Theorem 2, the two policy updates given in Equation 3.30 and Equation 3.37 respectively, must lead to the same resulting policy π'. Therefore, not surprisingly, we have an analogous policy improvement theorem in this case.

Theorem 4 $\Pi H\mathbf{r} \leq \Pi' H\mathbf{r}$ *implies* $M\Pi\mathbf{r} \leq M'\Pi'\mathbf{r}$

Proof: Plugging $M\Pi = \Pi H$ (see Lemma 11) and $H = (1-\gamma)I + \gamma P\Pi H$ (see Equation 3.25) into the assumption $\Pi H\mathbf{r} \leq \Pi' H\mathbf{r}$, we have

$$M\Pi\mathbf{r} \leq (1-\gamma)\Pi'\mathbf{r} + \gamma\Pi' P\Pi H\mathbf{r} \tag{3.40}$$

Using the assumption $\Pi H\mathbf{r} \leq \Pi' H\mathbf{r}$ and Equation 3.25 recursively in the above relation,

$$\begin{aligned}
(1-\gamma)\Pi'\mathbf{r} + \gamma\Pi' P\Pi H\mathbf{r} &\leq (1-\gamma)\Pi'\mathbf{r} + \gamma\Pi' P\Pi' H\mathbf{r} \\
&= (1-\gamma)\Pi'\mathbf{r} + \gamma\Pi' P\Pi' [(1-\gamma)I + \gamma P\Pi H]\mathbf{r} \\
&= (1-\gamma)\Pi'\mathbf{r} + (1-\gamma)\gamma\Pi' P\Pi'\mathbf{r} + \gamma^2(\Pi' P)^2\Pi H\mathbf{r} \\
&\leq (1-\gamma)\Pi'\mathbf{r} + (1-\gamma)\gamma\Pi' P\Pi'\mathbf{r} + \gamma^2(\Pi' P)^2\Pi' H\mathbf{r} \\
&\vdots \\
&= (1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\Pi' P)^i\Pi'\mathbf{r}
\end{aligned}$$

From the definition of M (see Equation 3.14), we have

$$(1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\Pi' P)^i\Pi'\mathbf{r} = M'\Pi'\mathbf{r} \tag{3.41}$$

Therefore,

$$M\Pi\mathbf{r} \leq M'\Pi'\mathbf{r} \tag{3.42}$$

Thus, a dual state policy iteration algorithm can be completely expressed in terms of the dual representation M, incorporating both dual policy evaluation (see Equation 3.15) and dual policy improvement (see Equation 3.38) leading to an equivalent result to the standard primal policy iteration algorithm based on $\mathbf{v}$ (see Equation 3.10) and primal policy improvement (see Equation 3.31).

3.6 Off-Policy Update

The off-policy update is prominent in dynamic programming (e.g., value iteration) and reinforcement learning algorithms (e.g., Q-learning). The off-policy update $\mathcal{M}$ is different from on-policy update $\mathcal{O}$ in that it is neither linear nor defined by any reference policy, but instead applies a *greedy max* update to the current estimates.

In this section, I only present the planning algorithms using the off-policy update and their convergence analysis. In particular, I show primal Bellman iteration algorithms with value function representations and dual ones with visit distribution representations. By sampling the model of an MDP (P and $\mathbf{r}$) in these planning algorithms, we can get their corresponding learning algorithms with point-wise updates (see Appendix B).

3.6.1 Primal Representations

In the primal case, Bellman iteration of state value function corresponds to the well known value iteration algorithm (see Table 2.2), which is based on the off-policy update $\mathcal{M}$ for state value function $\mathbf{v}$

$$\begin{aligned} \mathcal{M}\mathbf{v} &= \Pi^*[\mathbf{r} + \gamma P\mathbf{v}] \quad \text{where} \\ \Pi^*[\mathbf{r} + \gamma P\mathbf{v}]_{(s)} &= \max_a \mathbf{r}_{(sa)} + \gamma P_{(sa,:)}\mathbf{v} \end{aligned} \tag{3.43}$$

The goal of this greedy update is to bring the representation $\mathbf{v}$ closer to satisfying the optimal-policy Bellman equation $\mathbf{v} = \Pi^*[\mathbf{r} + \gamma P\mathbf{v}]$.

Similarly, the off-policy update operator $\mathcal{M}$ for state-action value function $\mathbf{q}$ is

defined as

$$\begin{aligned} \mathcal{M}\mathbf{q} &= \mathbf{r} + \gamma P \Pi^*[\mathbf{q}] \quad \text{where} \\ \Pi^*[\mathbf{q}]_{(s)} &= \max_a \mathbf{q}_{(sa)} \end{aligned} \tag{3.44}$$

The goal of this greedy update is to bring the representation $\mathbf{q}$ closer to satisfying the optimal-policy Bellman equation $\mathbf{q} = \mathbf{r} + \gamma P \Pi^*[\mathbf{q}]$.

3.6.2 Dual Representations

In the dual, Bellman iteration bypass the explicit representation of the value of a policy, and attempt to update the evaluation of the optimal policy implicitly. An analogous off-policy update $\mathcal{M}$ for the state visit distribution can be defined as

$$\begin{aligned} \mathcal{M}M &= (1-\gamma)I + \gamma \Pi^*_{\mathbf{r}}[M]P, \quad \text{where} \\ \Pi^*_{\mathbf{r}}[M]_{(s,s'a')} &= 1_{(s=s')} 1_{(a'=a^*(s))} \\ a^*(s) &= \operatorname*{argmax}_a P(sa,:) M \Pi \mathbf{r} \end{aligned} \tag{3.45}$$

where $1_{(s=s')}$ equals 1 when $s = s'$ and 0 when $s \neq s'$, and $1_{(a'=a^*(s))}$ equals 1 when $a' = a^*(s)'$ and 0 when $a' \neq a^*(s)$. The goal of this greedy update is to bring the representation M closer to satisfying the optimal-policy Bellman equation $M = (1-\gamma)I + \gamma \Pi^*_{\mathbf{r}}[PM]$.

Similarly, the max-policy update operator $\mathcal{M}$ for state-action visit distribution H is defined as

$$\begin{aligned} \mathcal{M}H &= (1-\gamma)I + \gamma P \Pi^*_{\mathbf{r}}[H], \quad \text{where} \\ \Pi^*_{\mathbf{r}}[H]_{(s,s'a')} &= 1_{(s=s')} 1_{(a'=a^*(s))} \\ a^*(s) &= \operatorname*{argmax}_a \sum_{(s''a'')} H_{(sa,s''a'')} \mathbf{r}_{(s''a'')} \end{aligned} \tag{3.46}$$

The goal of this greedy update is to bring the representation H closer to satisfying the optimal-policy Bellman equation $H = (1-\gamma)I + \gamma P \Pi^*_{\mathbf{r}}[H]$.

Note that a dual form of Bellman iteration algorithms need not refer to the primal value functions at all. Nevertheless, the off-policy update of $\mathbf{q}$ (see Equation

3.44) and the off-policy update of H (see Equation 3.46) behave equivalently because of their relation $(1-\gamma)\mathbf{q} = H\mathbf{r}$ (see Theorem 2 in Section 3.4.2).

Lemma 16 *If* $(1-\gamma)\mathbf{q} = H\mathbf{r}$, *then* $(1-\gamma)\mathcal{M}\mathbf{q} = \mathcal{M}H\mathbf{r}$.

Proof: From the assumption,

$$(1-\gamma)\mathbf{q} = H\mathbf{r} \implies (1-\gamma)\mathbf{q}_{(sa)} = [H\mathbf{r}]_{(sa)} \quad \forall (sa)$$

Together with the definitions of Π^* in Equation 3.44 and 3.46, we have

$$(1-\gamma)\Pi^*[\mathbf{q}] = \Pi^*[H\mathbf{r}] \implies (1-\gamma)\Pi^*[\mathbf{q}] = \Pi^*_{\mathbf{r}}[H]\mathbf{r}$$

Applying $\mathcal{M}$ update on $\mathbf{q}$ (see Equation 3.44), we get

$$(1-\gamma)\mathcal{M}\mathbf{q} = (1-\gamma)(\mathbf{r}+\gamma P\Pi^*[\mathbf{q}])$$

Substituting $(1-\gamma)\Pi^*[\mathbf{q}]$ with $\Pi^*_{\mathbf{r}}[H]\mathbf{r}$, we have

$$(1-\gamma)\mathcal{M}\mathbf{q} = (1-\gamma)\mathbf{r} + \gamma P\Pi^*_{\mathbf{r}}[H])\mathbf{r}$$

Note that $\mathcal{M}H = (1-\gamma)I + \gamma P\Pi^*_{\mathbf{r}}[H]$ (see Equation 3.46),

$$(1-\gamma)\mathcal{M}\mathbf{q} = \mathcal{M}H\mathbf{r}$$

∎

Similarly, the off-policy update of $\mathbf{v}$ (see Equation 3.43) and the update of M (see Equation 3.45) behave equivalently because of their relation $(1-\gamma)\mathbf{v} = M\Pi\mathbf{r}$ (see Theorem 1 in Section 3.4.1). That is, $(1-\gamma)\mathcal{M}\mathbf{v} = \mathcal{M}M\Pi\mathbf{r}$ holds.

3.6.3 Convergence Analysis

To keep the presentation efficient, here I only show the convergence analysis of off-policy update for the state-action based representations, $\mathbf{q}$ and H. Analogous results are easily obtained for the state-based representations, $\mathbf{v}$ and M. The strategy for establishing convergence for the nonlinear max operator is similar to the on-policy case, but involves working with a different norm. Instead of considering a 2-norm

weighted by the visit probabilities induced by a fixed policy, one simply uses the max-norm in this case

$$\|\mathbf{q}\|_\infty = \max_{(sa)} |q_{(sa)}| \tag{3.47}$$

The contraction property of the $\mathcal{M}$ operator with respect to this norm can then be established in the primal case as follows.

Theorem 5 $\|\mathcal{M}\mathbf{q}_1 - \mathcal{M}\mathbf{q}_2\|_\infty \leq \gamma \|\mathbf{q}_1 - \mathbf{q}_2\|_\infty$ *(Bertsekas, 1995)*

Proof: Let scalar δ

$$\delta = \|\mathbf{q}_1 - \mathbf{q}_2\|_\infty$$

Then

$$\mathbf{q}_2 - \delta\mathbf{e} \leq \mathbf{q}_1 \leq \mathbf{q}_2 + \delta\mathbf{e}$$

By Lemma 17 and Lemma 18 below, we have

$$\begin{aligned} \mathcal{M}\mathbf{q}_2 - \gamma\delta\mathbf{e} &\leq \mathcal{M}\mathbf{q}_1 \leq \mathcal{M}\mathbf{q}_2 + \gamma\delta\mathbf{e} \\ \Longrightarrow \mathcal{M}\mathbf{q}_1 - \mathcal{M}\mathbf{q}_2 \leq \gamma\delta\mathbf{e} &\text{ and } \mathcal{M}\mathbf{q}_1 - \mathcal{M}\mathbf{q}_2 \geq -\gamma\delta\mathbf{e} \\ \Longrightarrow \|\mathcal{M}\mathbf{q}_1 - \mathcal{M}\mathbf{q}_2\|_\infty &\leq \gamma\delta \end{aligned}$$

That is

$$\|\mathcal{M}\mathbf{q}_1 - \mathcal{M}\mathbf{q}_2\|_\infty \leq \gamma \|\mathbf{q}_1 - \mathbf{q}_2\|_\infty$$

∎

Lemma 17 $\mathcal{M}(\mathbf{q} + \delta\mathbf{e}) = \mathcal{M}\mathbf{q} + \gamma\delta\mathbf{e}$ *(Bertsekas, 1995)*

Proof:

$$\begin{aligned} \mathcal{M}(\mathbf{q} + \delta\mathbf{e}) &= \mathbf{r} + \gamma P \Pi^*[\mathbf{q} + \delta\mathbf{e}] \\ &= \mathbf{r} + \gamma P \Pi^*[\mathbf{q}] + \gamma\delta\mathbf{e} \\ &= \mathcal{M}\mathbf{q} + \gamma\delta\mathbf{e} \end{aligned}$$

∎

Lemma 18 *If* $\mathbf{q}_1 \leq \mathbf{q}_2$, *then* $\mathcal{M}\mathbf{q}_1 \leq \mathcal{M}\mathbf{q}_2$ *(Bertsekas, 1995).*

Proof:

$$\begin{aligned}
& \mathbf{q}_1 \leq \mathbf{q}_2 \\
\Longrightarrow\ & \mathbf{q}_{1(sa)} \leq \mathbf{q}_{2(sa)}\ \forall s, a \\
\Longrightarrow\ & \max_a \mathbf{q}_{1(sa)} \leq \max_a \mathbf{q}_{2(sa)} \\
\Longrightarrow\ & \Pi^*[\mathbf{q}_1] \leq \Pi^*[\mathbf{q}_2] \\
\Longrightarrow\ & \gamma P \Pi^*[\mathbf{q}_1] \leq \gamma P \Pi^*[\mathbf{q}_2] \\
\Longrightarrow\ & \mathbf{r} + \gamma P \Pi^*[\mathbf{q}_1] \leq \mathbf{r} + \gamma P \Pi^*[\mathbf{q}_2] \\
\Longrightarrow\ & \mathcal{M}\mathbf{q}_1 \leq \mathcal{M}\mathbf{q}_2
\end{aligned}$$

∎

The above theorem shows that contraction suffices to establish the existence of a unique fixed point of $\mathcal{M}$ among vectors $\mathbf{q}$, and that repeated application of $\mathcal{M}$ converges to a fixed point $\mathbf{q}_*$ such that $\mathcal{M}\mathbf{q}_* = \mathbf{q}_*$.

To establish convergence of the off-policy update in the dual representation, First I define the max-norm for state-action visit distribution as

$$\|H\|_\infty = \max_{(sa)} |\sum_{(s'a')} H_{(sa,s'a')}\mathbf{r}_{(s'a')}| \tag{3.48}$$

Then I will simply reduce the dual to the primal case by appealing to the relationship $(1-\gamma)\mathcal{M}\mathbf{q} = \mathcal{M}H\mathbf{r}$ (see Lemma 16) to prove convergence of $\mathcal{M}H$.

Given convergence of $\mathcal{M}\mathbf{q}$ to a fixed point $\mathcal{M}\mathbf{q}_* = \mathbf{q}_*$, the same must also hold for $\mathcal{M}H$. Again, one subtlety here is that the dual fixed point is not unique. That is, the relationship between H and $\mathbf{q}$ is many to one, and several matrices can correspond to the same $\mathbf{q}$. These matrices form a convex subspace (in fact, a simplex), since if $H_1\mathbf{r} = (1-\gamma)\mathbf{q}$ and $H_2\mathbf{r} = (1-\gamma)\mathbf{q}$ then $(\alpha H_1 + (1-\alpha)H_2)\mathbf{r} = (1-\gamma)\mathbf{q}$ for any α, where furthermore α must be restricted to $0 \leq \alpha \leq 1$ to maintain nonnegativity. The simplex of fixed points $\{H_* : \mathcal{M}H_* = H_*\}$ is given by matrices H_* that satisfy $H_*\mathbf{r} = (1-\gamma)\mathbf{q}_*$.

So far I presented novel, dual algorithms for dynamic programming in tabular case, based on maintaining explicit representations of stationary distributions

instead of value functions. Primal and dual representations (algorithms) exhibit strong equivalence in the tabular case, as they should. However, when I begin to consider approximation, differences emerge.

3.7 Approximate Dynamic Programming

Scaling up DP and RL algorithms to large sequential decision making problem domains has received a great deal of attention in recent years. One commonly used technique is to generalize across state or state-action space by exploiting the structure of the problem with function approximation. That is, the target function is approximated as a parameterized function of a set of basis functions (also usually referred to as "bases" or "features"). This parametrization could be either linear approximation or non-linear approximation.

3.7.1 Approximate Representations

Here I examine the representations of linear approximation for both the primal and dual cases since linear function approximation is widely used.

Primal Linear Approximation

In the primal case, linear approximation proceeds by fixing a small set of basis functions, forming a $|S||A| \times k$ matrix Φ, where k is the number of bases. The approximation of $\mathbf{q}$ can be expressed by a linear combination of bases

$$\hat{\mathbf{q}} = \Phi \mathbf{w} \tag{3.49}$$

where $\mathbf{w}$ is a $k \times 1$ vector of adjustable weights. This is equivalent to maintaining the constraint that $\hat{\mathbf{q}} \in \text{col_span}(\Phi)$.

Dual Linear Approximation

In the dual, a similar linear approximation strategy can be followed as in the primal case, except that constraints need to be added to ensure the resulting approximation $\hat{H}$ is *row* normalized. That is, we start with a linear representation

$$\hat{H} = reshape(\Psi \mathbf{w}) \tag{3.50}$$

where $\mathbf{w}$ is a $k \times 1$ vector of adjustable weights as it is in the primal case, Ψ is a $(|S||A|)^2 \times k$ matrix of basis functions, and the operator $reshape$ converts the vector $\Psi\mathbf{w}$, whose dimension is $(|S||A|)^2 \times 1$, into a $|S||A| \times |S||A|$ matrix. Since the vec operator[4] is an inverse operator of the $reshape$ operator, the above linear representation can be rewritten as

$$vec(\hat{H}) = \Psi\mathbf{w} \tag{3.51}$$

I explain this representation a little further. We know that matrix Ψ has k columns of bases. The i^{th} column of Ψ can then be viewed as $\Psi_{(:,i)} = vec(\Upsilon^{(i)})$, where $\Upsilon^{(i)}$ is an $|S||A| \times |S||A|$ matrix. Each of these basis matrices is required to be row normalized (i.e, $\Upsilon^{(i)}\mathbf{1} = \mathbf{1}$). Since $\Upsilon^{(i)}\mathbf{1}$ is a $|S||A| \times 1$ vector, this constraint can be expressed by

$$\Upsilon^{(i)}\mathbf{1} = vec(\Upsilon^{(i)}\mathbf{1}) \tag{3.52}$$

Using $vec(\mathbf{AB}) = (\mathbf{B}^\top \otimes I)vec(\mathbf{A})$, [5] it follows that

$$vec(\Upsilon^{(i)}\mathbf{1}) = (\mathbf{1}^\top \otimes I)vec(\Upsilon^{(i)}) = (\mathbf{1}^\top \otimes I)\Psi_{(:,i)} \tag{3.53}$$

where the operator $\otimes$ is the Kronecker product.[6]

---

[4]The vec operator creates a column vector from a matrix $\mathbf{A}$ by stacking the column vectors of $\mathbf{A} = (\mathbf{a}_1\mathbf{a}_2\cdots\mathbf{a}_n)$ below one another:

$$vec(\mathbf{A}) = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{pmatrix}.$$

[5]Recall Proposition 7.1.9. in Book (Bernstein, 2005).

[6]Let $\mathbf{A}$ be an $n \times p$ matrix and $\mathbf{B}$ be an $m \times q$ matrix. The $mn \times pq$ matrix

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{(11)}\mathbf{B} & a_{(12)}\mathbf{B} & \cdots & a_{(1p)}\mathbf{B} \\ a_{(21)}\mathbf{B} & a_{(22)}\mathbf{B} & \cdots & a_{(2p)}\mathbf{B} \\ \vdots & \vdots & \vdots & \vdots \\ a_{(n1)}\mathbf{B} & a_{(n2)}\mathbf{B} & \cdots & a_{(np)}\mathbf{B} \end{pmatrix}$$

is called the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$. It is also called the direct product or the tensor product.

Therefore, we have

$$\begin{aligned}
(\mathbf{1}^\top \otimes I)\Psi &= (\mathbf{1}^\top \otimes I)[\Psi_{(:,1)} \cdots \Psi_{(:,i)}) \cdots \Psi_{(:,k))}] \\
&= [(\mathbf{1}^\top \otimes I)\Psi_{(:,1)} \cdots (\mathbf{1}^\top \otimes I)\Psi_{(:,i))} \cdots (\mathbf{1}^\top \otimes I)\Psi_{(:,k))}] \\
&= [(\mathbf{1}^\top \otimes I)vec(\Upsilon^{(1)}) \cdots (\mathbf{1}^\top \otimes I)vec(\Upsilon^{(i)}) \cdots (\mathbf{1}^\top \otimes I)vec(\Upsilon^{(k)})] \\
&= [vec(\Upsilon^{(1)}\mathbf{1}) \cdots vec(\Upsilon^{(i)}\mathbf{1}) \cdots vec(\Upsilon^{(k)}\mathbf{1})] \\
&= [\mathbf{1} \cdots \mathbf{1} \cdots \mathbf{1}] \\
&= \mathbf{1}\mathbf{1}^\top
\end{aligned} \tag{3.54}$$

where $\mathbf{1}\mathbf{1}^\top$ defines a $|S||A| \times k$ matrix of all ones.

To ensure that $\hat{H}$ remains a nonnegative, row normalized approximation to H, we simply add the constraints

$$\Psi \geq 0, \quad (\mathbf{1}^\top \otimes I)\Psi = \mathbf{1}\mathbf{1}^\top \tag{3.55}$$

and

$$\mathbf{w} \geq 0, \quad \mathbf{w}^\top \mathbf{1} = 1 \tag{3.56}$$

Lemma 19 *If* $\Psi \geq 0$, $(\mathbf{1}^\top \otimes I)\Psi = \mathbf{1}\mathbf{1}^\top$ *and* $\mathbf{w} \geq 0$, $\mathbf{w}^\top \mathbf{1} = 1$, *then* $vec(\hat{H}) = \Psi\mathbf{w}$ *satisfies* $\hat{H} \geq 0$ *and* $\hat{H}\mathbf{1} = \mathbf{1}$.

Proof: From the assumption that $\Psi \geq 0$ and $\mathbf{w} \geq 0$, $\hat{H} \geq 0$ is obvious.
From the assumption that $(\mathbf{1}^\top \otimes I)\Psi = \mathbf{1}\mathbf{1}^\top$ and $\mathbf{w}^\top \mathbf{1} = 1$, we have

$$(\mathbf{1}^\top \otimes I)\Psi\mathbf{w} = \mathbf{1}\mathbf{1}^\top\mathbf{w} = \mathbf{1} \tag{3.57}$$

Plugging $vec(\hat{H}) = \Psi\mathbf{w}$ into Equation 3.57, we have

$$(\mathbf{1}^\top \otimes I)vec(\hat{H}) = \mathbf{1}$$

Since $vec(\mathbf{AB}) = (\mathbf{B}^\top \otimes I)vec(\mathbf{A})$, we know that

$$vec(\hat{H}\mathbf{1}) = \mathbf{1} \tag{3.58}$$

Since $\hat{H}\mathbf{1}$ is an $|S||A| \times 1$ vector,

$$\hat{H}\mathbf{1} = \mathbf{1} \tag{3.59}$$

In summary, in the dual the linear approximation of matrix H is

$$vec(\hat{H}) = \Psi \mathbf{w} \quad \text{subject to}$$
$$\Psi \geq 0, \ (\mathbf{1}^\top \otimes I)\Psi = \mathbf{1}\mathbf{1}^\top, \ \mathbf{w} \geq 0, \ \mathbf{w}^\top \mathbf{1} = \mathbf{1} \tag{3.60}$$

I denote the constraint on $\hat{H}$ as $\hat{H} \in \text{simplex}(\Psi)$, such that $\text{simplex}(\Psi) \equiv \{\hat{H} : vec(\hat{H}) = \Psi \mathbf{w}, \Psi \geq 0, \ (\mathbf{1}^\top \otimes I)\Psi = \mathbf{1}\mathbf{1}^\top, \ \mathbf{w} \geq 0, \ \mathbf{w}^\top \mathbf{1} = \mathbf{1}\}$. This means that the bases can be reshaped into row normalized, nonnegative matrices. Therefore, it is sufficient for the weight vector $\mathbf{w}$ to be positive and normalized to ensure that H is positive and row normalized.

In this section, I first introduce operators (projection operator and gradient operator) that ensure the approximations stay representable in the given bases. Then I consider their composition with the on-policy update and off-policy update, and analyze their convergence properties. For the composition of the on-policy update and projection operator, I establish a similar bound on approximation error in the dual case as in the primal case. Famously, the off-policy update does not always have a fixed point when combined with approximation in the primal case (de Farias & Van Roy, 2000), and consequently suffers the risk of divergence (Baird, 1995; Sutton & Barto, 1998). However, in the dual case the off-policy update with linear approximation cannot diverge because of boundedness of the dual representations; although the question of whether off-policy update always converges in the dual remains open.

3.7.2 Projection Operator

Recall that in the primal, the action value function $\mathbf{q}$ is approximated by a linear combination of bases in Φ (see Equation 3.49). Unfortunately, there is no reason to expect $\mathcal{O}\mathbf{q}$ or $\mathcal{M}\mathbf{q}$ to stay in the column span of Φ, so a best approximation is required. The subtlety resolved by Tsitsiklis and Van Roy (1997) is to identify a particular form of best approximation—weighted least squares—that ensures convergence is still achieved when combined with the on-policy operator $\mathcal{O}$. Unfortunately, the fixed point of this combined update operator is not guaranteed to be

the best representable approximation of $\mathcal{O}$'s fixed point, $\mathbf{q}_\Pi$. Nevertheless, a bound can be proved on how close this altered fixed point is to the best representable approximation.

Primal Case

We summarize a few details that will be useful below. First, the best least squares approximation is computed with respect to the distribution $\mathbf{z}$. The map from a general $\mathbf{q}$ vector onto its best approximation in col_span(Φ) is defined by another operator, $\mathcal{P}$, which projects $\mathbf{q}$ into the column span of Φ

$$\mathcal{P}\mathbf{q} = \underset{\hat{\mathbf{q}} \in \text{col_span}(\Phi)}{\text{argmin}} \|\mathbf{q} - \hat{\mathbf{q}}\|_{\mathbf{z}}^2 \tag{3.61}$$

where $\mathbf{q}$ is the true value function and $\hat{\mathbf{q}}$ is an approximation for it.

The above equation can be rewritten as

$$\mathcal{P}\mathbf{q} = \underset{\hat{\mathbf{q}}}{\text{argmin}} \|\mathbf{q} - \hat{\mathbf{q}}\|_{\mathbf{z}}^2 \quad \text{subject to } \hat{\mathbf{q}} = \Phi\mathbf{w} \text{ for some } \mathbf{w} \tag{3.62}$$

This optimization problem can be expressed in terms of $\mathbf{w}$ as

$$\mathcal{P}\mathbf{q} = \Phi\mathbf{w}^* \quad \text{subject to } \mathbf{w}^* = \underset{\mathbf{w}}{\text{argmin}} \|\mathbf{q} - \Phi\mathbf{w}\|_{\mathbf{z}}^2 \tag{3.63}$$

$\mathbf{w}^*$ can be solved by taking derivative of $\|\mathbf{q} - \Phi\mathbf{w}\|_{\mathbf{z}}^2$ with respect to $\mathbf{w}$ and letting it equal to zero,

$$\mathbf{w}^* = (\Phi^\top Z \Phi)^{-1} \Phi^\top Z \mathbf{q} \tag{3.64}$$

Plugging $\mathbf{w}^*$ back into Equation 3.63, we have

$$\mathcal{P}\mathbf{q} = \Phi(\Phi^\top Z \Phi)^{-1} \Phi^\top Z \mathbf{q}$$

The important property of this weighted projection is that it is a non-expansion operator in $\|\cdot\|_{\mathbf{z}}$, i.e., $\|\mathcal{P}\mathbf{q}\|_{\mathbf{z}} \leq \|\mathbf{q}\|_{\mathbf{z}}$, which can be easily obtained from the generalized Phythagorean theorem (see below Lemma 20).

Lemma 20 $\|\mathbf{q}\|_{\mathbf{z}}^2 = \|\mathcal{P}\mathbf{q}\|_{\mathbf{z}}^2 + \|\mathbf{q} - \mathcal{P}\mathbf{q}\|_{\mathbf{z}}^2$ *(Tsitsiklis & Van Roy, 1997)*

Approximate dynamic programming then proceeds by composing the two operators —the on-policy update $\mathcal{O}$ with the subspace projection $\mathcal{P}$— essentially computing the best representable approximation of the one step update. This combined operator is guaranteed to converge, since composing a non-expansion with a contraction is still a contraction. In fact, Lemma 13 can be re-established for the composition $\mathcal{PO}$. Thus, by the contraction map fixed point theorem (Bertsekas, 1995) a fixed point must exist and is unique. Let $\mathbf{q}_+ = \mathcal{PO}\mathbf{q}_+$ be the fixed point of the combined operator. Unfortunately, the fixed point $\mathbf{q}_+$ is not guaranteed to be the best representable approximation of $\mathcal{O}$'s fixed point $\mathbf{q}_\Pi$. Nevertheless, a bound can be proved on how close the altered fixed point $\mathbf{q}_+$ is to the best representable approximation $\mathcal{P}\mathbf{q}_\Pi$ of $\mathcal{O}$'s fixed point.

Lemma 21 $\|\mathbf{q}_+ - \mathbf{q}_\Pi\|_\mathbf{z} \leq \frac{1}{1-\gamma}\|\mathcal{P}\mathbf{q}_\Pi - \mathbf{q}_\Pi\|_\mathbf{z}$ *(Tsitsiklis & Van Roy, 1997)*

Proof: First note that

$$\begin{aligned} \|\mathbf{q}_+ - \mathbf{q}_\Pi\|_\mathbf{z} &= \|\mathbf{q}_+ - \mathcal{P}\mathbf{q}_\Pi + \mathcal{P}\mathbf{q}_\Pi - \mathbf{q}_\Pi\|_\mathbf{z} \\ &\leq \|\mathbf{q}_+ - \mathcal{P}\mathbf{q}_\Pi\|_\mathbf{z} + \|\mathcal{P}\mathbf{q}_\Pi - \mathbf{q}_\Pi\|_\mathbf{z} \end{aligned} \tag{3.65}$$

Since $\mathbf{q}_+ = \mathcal{PO}\mathbf{q}_+$, we have

$$\|\mathbf{q}_+ - \mathcal{P}\mathbf{q}_\Pi\|_\mathbf{z} = \|\mathcal{PO}\mathbf{q}_+ - \mathcal{P}\mathbf{q}_\Pi\|_\mathbf{z} \tag{3.66}$$

Next notice that $\mathcal{P}$ is a non-expansion operator and $\mathbf{q}_\Pi = \mathcal{O}\mathbf{q}_\Pi$,

$$\|\mathcal{PO}\mathbf{q}_+ - \mathcal{P}\mathbf{q}_\Pi\|_\mathbf{z} \leq \|\mathcal{O}\mathbf{q}_+ - \mathbf{q}_\Pi\|_\mathbf{z} = \|\mathcal{O}\mathbf{q}_+ - \mathcal{O}\mathbf{q}_\Pi\|_\mathbf{z} \tag{3.67}$$

By Lemma 13, we also have

$$\|\mathcal{O}\mathbf{q}_+ - \mathcal{O}\mathbf{q}_\Pi\|_\mathbf{z} \leq \gamma\|\mathbf{q}_+ - \mathbf{q}_\Pi\|_\mathbf{z} \tag{3.68}$$

From Equations 3.66, 3.67, and 3.68, we have

$$\|\mathbf{q}_+ - \mathcal{P}\mathbf{q}_\Pi\|_\mathbf{z} \leq \gamma\|\mathbf{q}_+ - \mathbf{q}_\Pi\|_\mathbf{z} \tag{3.69}$$

Plugging Equation 3.69 back into Equation 3.65, then we have

$$\|\mathbf{q}_+ - \mathbf{q}_\Pi\|_\mathbf{z} \leq \gamma\|\mathbf{q}_+ - \mathbf{q}_\Pi\|_\mathbf{z} + \|\mathcal{P}\mathbf{q}_\Pi - \mathbf{q}_\Pi\|_\mathbf{z}$$

Since $0 \leq \gamma < 1$, rearranging the above equation yields,

$$\|\mathbf{q}_+ - \mathbf{q}_\Pi\|_\mathbf{z} \leq \frac{1}{1-\gamma}\|\mathcal{P}\mathbf{q}_\Pi - \mathbf{q}_\Pi\|_\mathbf{z}$$

■

Linear function approximation in the dual case is a bit more complicated because matrices are being represented, not vectors, and moreover the matrices need to satisfy row normalization and nonnegativity constraints. Nevertheless, a very similar approach to the primal case can be successfully applied.

Dual Case

Recall that in the dual, state-action visit distribution H is approximated by a linear combination of bases in Ψ (see Equation 3.60). As in the primal case, there is no reason to expect that an update like $\mathcal{O}H$ should keep the matrix in the simplex. Therefore, a projection operator must be constructed that determines the best representable approximation to $\mathcal{O}H$. One needs to be careful to define this projection with respect to the right norm to ensure convergence. Here the pseudo-norm $\|\cdot\|_{\mathbf{z},\mathbf{r}}$ defined in Equation 3.29 suits this purpose. Define the weighted projection operator $\mathcal{P}$ over matrices

$$\mathcal{P}H = \underset{\hat{H} \in \text{simplex}(\Psi)}{\text{argmin}} \|H - \hat{H}\|_{\mathbf{z},\mathbf{r}}^2 \tag{3.70}$$

The projection could be obtained by solving the above quadratic program. A key result is that this projection operator is a non-expansion with respect to the pseudo-norm $\|\cdot\|_{\mathbf{z},\mathbf{r}}$.

Theorem 6 $\|\mathcal{P}H\|_{\mathbf{z},\mathbf{r}} \leq \|H\|_{\mathbf{z},\mathbf{r}}$

Proof: The easiest way to prove the theorem is to observe that the projection operator $\mathcal{P}$ is really a composition of three orthogonal projections: first, onto the linear subspace span(Ψ), then onto the subspace of row normalized matrices span(Ψ) $\cap$ $\{H : H\mathbf{1} = \mathbf{1}\}$, and finally onto the space of nonnegative matrices span(Ψ) $\cap$ $\{H : H\mathbf{1} = \mathbf{1}\} \cap \{H : H \geq 0\}$. Note that the last projection into the nonnegative

halfspace is equivalent to a projection into a linear subspace for some hyperplane tangent to the simplex.

Each one of these projections is a non-expansion in $\|\cdot\|_{\mathbf{z},\mathbf{r}}$ in the same way: a generalized Pythagorean theorem holds. Consider just one of these linear projections $\mathcal{P}_1$.

$$\begin{aligned}
\|H\|_{\mathbf{z},\mathbf{r}}{}^2 &= \|\mathcal{P}_1 H + H - \mathcal{P}_1 H\|_{\mathbf{z},\mathbf{r}}{}^2 \\
&= \|\mathcal{P}_1 H\mathbf{r} + H\mathbf{r} - \mathcal{P}_1 H\mathbf{r}\|_{\mathbf{z}}{}^2 \\
&= \|\mathcal{P}_1 H\mathbf{r}\|_{\mathbf{z}}{}^2 + \|H\mathbf{r} - \mathcal{P}_1 H\mathbf{r}\|_{\mathbf{z}}{}^2 \\
&= \|\mathcal{P}_1 H\|_{\mathbf{z},\mathbf{r}}{}^2 + \|H - \mathcal{P}_1\|_{\mathbf{z},\mathbf{r}}{}^2
\end{aligned}$$

where the third equality follows from Lemma 20. Since the overall projection is just a composition of non-expansions, it too must be a non-expansion. ∎

As in the primal, approximate dynamic programming can be implemented by composing the on-policy update $\mathcal{O}$ with the projection operator $\mathcal{P}$. Since $\mathcal{O}$ is a contraction and $\mathcal{P}$ a non-expansion, $\mathcal{P}\mathcal{O}$ must also be a contraction, and it then follows that it has a fixed point. Note that, as in the tabular case, this fixed point is only unique up to $H\mathbf{r}$-equivalence, since the pseudo-norm $\|\cdot\|_{\mathbf{z},\mathbf{r}}$ does not distinguish H_1 and H_2 such that $H_1\mathbf{r} = H_2\mathbf{r}$. Here too, the fixed point is actually a simplex of equivalent solutions. For simplicity, I denote the simplex of fixed points for $\mathcal{P}\mathcal{O}$ by some representative $H_+ = \mathcal{P}\mathcal{O}H_+$.

Finally, I can recover an approximation bound that is analogous to the primal bound, which bounds the approximation error between H_+ and the best representable approximation to the on-policy fixed point $H_\Pi = \mathcal{O}H_\Pi$.

Theorem 7 $\|H_+ - H_\Pi\|_{\mathbf{z},\mathbf{r}} \leq \frac{1}{1-\gamma}\|\mathcal{P}H_\Pi - H_\Pi\|_{\mathbf{z},\mathbf{r}}$

Proof: First note that

$$\begin{aligned}
\|H_+ - H_\Pi\|_{\mathbf{z},\mathbf{r}} &= \|H_+ - \mathcal{P}H_\Pi + \mathcal{P}H_\Pi - H_\Pi\|_{\mathbf{z},\mathbf{r}} \\
&\leq \|H_+ - \mathcal{P}H_\Pi\|_{\mathbf{z},\mathbf{r}} + \|\mathcal{P}H_\Pi - H_\Pi\|_{\mathbf{z},\mathbf{r}} \qquad (3.71)
\end{aligned}$$

Since $H_+ = \mathcal{P}\mathcal{O}H_+$, we have

$$\|H_+ - \mathcal{P}H_\Pi\|_{\mathbf{z},\mathbf{r}} = \|\mathcal{P}\mathcal{O}H_+ - \mathcal{P}H_\Pi\|_{\mathbf{z},\mathbf{r}} \qquad (3.72)$$

Next notice that $\mathcal{P}$ is a non-expansion operator and $H_\Pi = \mathcal{O}H_\Pi$,

$$\|\mathcal{P}\mathcal{O}H_+ - \mathcal{P}H_\Pi\|_{\mathbf{z},\mathbf{r}} \leq \|\mathcal{O}H_+ - H_\Pi\|_{\mathbf{z},\mathbf{r}} = \|\mathcal{O}H_+ - \mathcal{O}H_\Pi\|_{\mathbf{z},\mathbf{r}} \quad (3.73)$$

By Lemma 15, we also have

$$\|\mathcal{O}H_+ - \mathcal{O}H_\Pi\|_{\mathbf{z},\mathbf{r}} \leq \gamma\|H_+ - H_\Pi\|_{\mathbf{z},\mathbf{r}} \quad (3.74)$$

From Equations 3.72, 3.73, and 3.74, we have

$$\|H_+ - \mathcal{P}H_\Pi\|_{\mathbf{z},\mathbf{r}} \leq \gamma\|H_+ - H_\Pi\|_{\mathbf{z},\mathbf{r}} \quad (3.75)$$

Plugging Equation 3.75 back into Equation 3.71, then we have

$$\|H_+ - H_\Pi\|_{\mathbf{z},\mathbf{r}} \leq \gamma\|H_+ - H_\Pi\|_{\mathbf{z},\mathbf{r}} + \|\mathcal{P}H_\Pi - H_\Pi\|_{\mathbf{z},\mathbf{r}}$$

Since $0 \leq \gamma < 1$, rearranging the above equation yields,

$$\|H_+ - H_\Pi\|_{\mathbf{z},\mathbf{r}} \leq \frac{1}{1-\gamma}\|\mathcal{P}H_\Pi - H_\Pi\|_{\mathbf{z},\mathbf{r}}$$

■

To compare the primal and dual results, note that despite the similarity of the bounds, the projection operators do not preserve the tight relationship between primal and dual updates. That is, even if $(1-\gamma)\mathbf{q} = H\mathbf{r}$ and $(1-\gamma)(\mathcal{O}\mathbf{q}) = (\mathcal{O}H)\mathbf{r}$, it is not true in general that $(1-\gamma)(\mathcal{P}\mathcal{O}\mathbf{q}) = (\mathcal{P}\mathcal{O}H)\mathbf{r}$. The most obvious difference comes from the fact that in the dual, the space of H matrices has bounded diameter, whereas in the primal, the space of $\mathbf{q}$ vectors has unbounded diameter in the natural norms. Automatically, the dual updates cannot diverge with compositions like $\mathcal{P}\mathcal{O}$ and $\mathcal{P}\mathcal{M}$. However update $\mathcal{P}\mathcal{M}$ is known to not have fixed points in the primal in general (de Farias & Van Roy, 2000).

3.7.3 Gradient Operator

In large scale problems one does not normally have the luxury of computing full dynamic programming updates that evaluate complete expectations over the entire domain, which requires knowing the stationary visit distribution $\mathbf{z}$ for $P\Pi$, i.e.,

knowing the model of an MDP. Moreover, full least squares projections are usually not practical to compute either. The main intermediate step toward practical algorithms is to formulate gradient step operators that only approximate complete projections. Conveniently, the gradient update and projection operators are independent of the on-policy and off-policy updates and can be applied in either case. However, as we will see below, the gradient update operator causes significant instability in the off-policy update, to the degree that divergence is a common phenomenon (much more so than with full projections). Composing approximation with off-policy update (max operator) in the primal case can be dangerous. All other operator combinations are much better behaved in practice, and even those that are not known to converge usually behave reasonably.

Primal Case

Gradient step updates are easily derived from a given projection operator. In this case, one always works directly with weight vectors $\mathbf{w}$, rather than $\mathbf{q}$ vectors. Recall that the projection operator actually is equivalent to solving for a vector $\mathbf{w}$ of basis combination weights that minimizes the least square objective

$$J_{\mathbf{q}} = \frac{1}{2}\|\mathbf{q} - \hat{\mathbf{q}}\|_{\mathbf{z}}^2 = \frac{1}{2}\|\mathbf{q} - \Phi\mathbf{w}\|_{\mathbf{z}}^2 \tag{3.76}$$

where $\hat{\mathbf{q}} = \Phi\mathbf{w}$.

The gradient of above objective with respect to $\mathbf{w}$ is

$$\nabla_{\mathbf{w}} J_{\mathbf{q}} = \Phi^\top Z(\Phi\mathbf{w} - \mathbf{q}) = \Phi^\top Z(\hat{\mathbf{q}} - \mathbf{q}) \tag{3.77}$$

Using the relation $\hat{\mathbf{q}} = \Phi\mathbf{w}$, we can derive the gradient update with respect to $\hat{\mathbf{q}}$ as

$$\mathcal{G}_{\hat{\mathbf{q}}}\mathbf{q} = \hat{\mathbf{q}} - \alpha\Phi\nabla_{\mathbf{w}} J_{\mathbf{q}} = \hat{\mathbf{q}} - \alpha\Phi\Phi^\top Z(\hat{\mathbf{q}} - \mathbf{q}) \tag{3.78}$$

where α is a positive step-size parameter.

Since the target vector $\mathbf{q}$ is determined by the underlying dynamic programming update, this gives the composed updates

$$\mathcal{G}_{\hat{\mathbf{q}}}\mathcal{O}\mathbf{q} = \hat{\mathbf{q}} - \alpha\Phi\Phi^\top Z(\hat{\mathbf{q}} - \mathcal{O}\mathbf{q}) \tag{3.79}$$

and

$$\mathcal{G}_{\hat{\mathbf{q}}}\mathcal{M}\mathbf{q} = \hat{\mathbf{q}} - \alpha\Phi\Phi^\top(\hat{\mathbf{q}} - \mathcal{M}\mathbf{q}) \tag{3.80}$$

respectively for the on-policy and off-policy cases.

In practice, an iterative optimization of $\hat{\mathbf{q}}$ is attempted by

$$\hat{\mathbf{q}}_{t+1} = \mathcal{G}_{\hat{\mathbf{q}}_t}\mathcal{O}\hat{\mathbf{q}}_t \tag{3.81}$$

and

$$\hat{\mathbf{q}}_{t+1} = \mathcal{G}_{\hat{\mathbf{q}}_t}\mathcal{M}\hat{\mathbf{q}}_t \tag{3.82}$$

respectively for the on-policy and off-policy cases.

Dual Case

In the dual representation, one can derive a gradient update operator in a similar way, except that it is important to maintain the constraints on the parameters w given the basis functions are the probability distributions. As in the primal case, I start by considering the projection objective

$$\begin{aligned} J_H &= \frac{1}{2}\|H - \hat{H}\|_{\mathbf{z},\mathbf{r}}^2 \text{ subject to} \\ vec(\hat{H}) &= \Psi\mathbf{w}, \ \ \mathbf{w} \geq 0, \ \ \mathbf{w}^\top\mathbf{1} = \mathbf{1} \end{aligned} \tag{3.83}$$

By the norm definition (see Equation 3.29), the above objective can be written as

$$J_H = \frac{1}{2}\|H\mathbf{r} - \hat{H}\mathbf{r}\|_{\mathbf{z}}^2 \tag{3.84}$$

Since $H\mathbf{r}$ and $\hat{H}\mathbf{r}$ are column vectors and $vec(\mathbf{AB}) = (\mathbf{B}^\top \otimes I)vec(\mathbf{A})$, we have

$$H\mathbf{r} = vec(H\mathbf{r}) = (\mathbf{r}^\top \otimes I)vec(H) = (\mathbf{r}^\top \otimes I)h \tag{3.85}$$

$$\hat{H}\mathbf{r} = vec(\hat{H}\mathbf{r}) = (\mathbf{r}^\top \otimes I)vec(\hat{H}) = (\mathbf{r}^\top \otimes I)\hat{h} \tag{3.86}$$

where $h \triangleq vec(H)$ and $\hat{h} \triangleq vec(\hat{H})$. Therefore, the objective becomes

$$\begin{aligned} J_H &= \frac{1}{2}\|vec(H\mathbf{r}) - vec(\hat{H}\mathbf{r})\|_{\mathbf{z}}^2 \\ &= \frac{1}{2}\|(\mathbf{r}^\top \otimes I)(vec(H) - vec(\hat{H}))\|_{\mathbf{z}}^2 \\ &= \frac{1}{2}\|(\mathbf{r}^\top \otimes I)(h - \Psi\mathbf{w})\|_{\mathbf{z}}^2 \end{aligned} \tag{3.87}$$

The unconstrained gradient of the above objective with respect to $\mathbf{w}$ is

$$\begin{aligned}\nabla_{\mathbf{w}} J_H &= \Psi^\top(\mathbf{r}^\top \otimes I)^\top Z(\mathbf{r}^\top \otimes I)(\Psi\mathbf{w} - h) \\ &= \Psi^\top(\mathbf{r}^\top \otimes I)^\top Z(\mathbf{r}^\top \otimes I)(\hat{h} - h) \\ &= \Gamma^\top Z(\mathbf{r}^\top \otimes I)(\hat{h} - h) \qquad (3.88)\end{aligned}$$

where $\Gamma = (\mathbf{r}^\top \otimes I)\Psi$, [7] and $diag(Z)$ corresponds to the stationary distribution over state-action pairs.

However, this gradient step cannot be followed directly because I need to maintain the constraints. The constraint $\mathbf{w}^\top 1 = 1$ can be maintained by first projecting the gradient onto it, obtaining

$$\delta\mathbf{w} = (I - \frac{1}{k}\mathbf{1}\mathbf{1}^\top)\nabla_{\mathbf{w}} J_H \qquad (3.89)$$

Thus, the weight vector can be updated by

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \alpha\delta\mathbf{w} \qquad (3.90) \\ &= \mathbf{w}_t - \alpha(I - \frac{1}{k}\mathbf{1}\mathbf{1}^\top)\Gamma^\top Z(\mathbf{r}^\top \otimes I)(\hat{h} - h) \qquad (3.91)\end{aligned}$$

where α is a step-size parameter.

Then the gradient operator can then be defined by

$$\begin{aligned}\mathcal{G}_{\hat{h}} h &= \hat{h} - \alpha\Psi\delta\mathbf{w} \qquad (3.92) \\ &= \hat{h} - \alpha\Psi(I - \frac{1}{k}\mathbf{1}\mathbf{1}^\top)\Gamma^\top Z(\mathbf{r}^\top \otimes I)(\hat{h} - h) \qquad (3.93)\end{aligned}$$

Similarly as in the primal, since the target vector H (i.e., h) is determined by the underlying dynamic programming update, this gives the composed updates

$$\mathcal{G}_{\hat{h}}\mathcal{O}h = \alpha\Psi(I - \frac{1}{k}\mathbf{1}\mathbf{1}^\top)\Gamma^\top Z(\mathbf{r}^\top \otimes I)(\hat{h} - \mathcal{O}h) \qquad (3.94)$$

[7] Note that Γ, whose dimension is $|S||A| \times k$, can be precomputed if $\mathbf{r}$ is known.

$$\begin{aligned}\Gamma &= (\mathbf{r}^\top \otimes I)[\Psi_{(:,1)} \cdots \Psi_{(:,i)}) \cdots \Psi_{(:,k))}] \\ &= [(\mathbf{r}^\top \otimes I)\Psi_{(:,1)} \cdots (\mathbf{r}^\top \otimes I)\Psi_{(:,i))} \cdots (\mathbf{r}^\top \otimes I)\Psi_{(:,k))}] \\ &= [(\mathbf{r}^\top \otimes I)vec(\Upsilon^{(1)}) \cdots (\mathbf{r}^\top \otimes I)vec(\Upsilon^{(i)}) \cdots (\mathbf{r}^\top \otimes I)vec(\Upsilon^{(k)})] \\ &= [\Upsilon^{(1)}\mathbf{r} \cdots \Upsilon^{(i)}\mathbf{r} \cdots \Upsilon^{(k)}\mathbf{r}]\end{aligned}$$

and

$$\mathcal{G}_{\hat{h}}\mathcal{M}h = \alpha\Psi(I - \frac{1}{k}\mathbf{1}\mathbf{1}^\top)\Gamma^\top(\mathbf{r}^\top \otimes I)(\hat{h} - \mathcal{M}h) \tag{3.95}$$

respectively for the on-policy and off-policy cases.

Again, iterations that attempt to optimize $\hat{h}$ in the on-policy and off-policy cases respectively are given by

$$\hat{h}_{t+1} = \mathcal{G}_{\hat{h}_t}\mathcal{O}\hat{h}_t \tag{3.96}$$

and

$$\hat{h}_{t+1} = \mathcal{G}_{\hat{h}_t}\mathcal{M}\hat{h}_t \tag{3.97}$$

The convergence properties of the gradient operators are investigated through the experimental studies in Section 3.8. The convergence of the $\mathcal{GO}$ operator is observed in both primal and dual cases for all the testing tasks. However, the $\mathcal{GM}$ operator behaves quite differently. In the primal, the divergence of the $\mathcal{GM}$ operator is observed in all the testing tasks while the $\mathcal{GM}$ operator always converges in the dual. The intuition of the convergence of the $\mathcal{GM}$ operator in the dual case is that the dual representations are bounded. Dual approximate reinforcement learning algorithms can be found in Appendix C.

So far, I have shown the dual dynamic programming algorithms with linear function approximation. The dual approach appears to hold a significant advantage over the standard primal approach: dual updates cannot diverge because the fundamental objects being represented are normalized probability distributions (i.e., belong to a bounded simplex).

3.8 Experimental Results

To investigate the effectiveness of the dual representations, I conducted experiments of the dynamic programming algorithms on randomly synthesized MDPs, on the *star problem*, and on the *mountain car* problem. The randomly synthesized MDP domains allow me to test the general properties of the algorithms. The star problem is perhaps the most-cited example of a problem where Q-learning with linear

function approximation diverges (Baird, 1995), and the mountain car domain has been prone to divergence with some primal representations (Boyan & Moore, 1995) although successful results were reported when bases are selected by sparse tile coding (Sutton, 1996).

I believe understanding the convergence properties of the dynamic programming algorithms will help understand the convergence of the reinforcement learning algorithms, which can be viewed as the sampled versions of the DP algorithms without the knowledge of an environment model. For each problem domain, twelve dynamic programming algorithms [8] with a finite horizon 1000 were run over 100 repeats. The discount factor was set to $\gamma = 0.9$. For on-policy algorithms, I measure the difference between the values generated by the algorithms and those generated by the analytically determined fixed-point. For off-policy algorithms, I measure the difference between the values generated by the resulting policy and the values of the optimal policy. The step size for the gradient updates was 0.1 for primal representations and 100 for dual representations. The initial values of state-action value functions $\mathbf{q}$ are set according to standard normal distribution and state-action visit distributions H are chosen uniformly randomly with row normalization.

In all the plots in this section, the x-axis is the number of the step in a sequential decision making process and the y-axis measures the error difference between different algorithms with respect to some particular norms. For the on-policy $\mathcal{O}$ update, the plots show the distance from the current estimates (either $\mathbf{q}$ or H) to the fixed point determined by the policy. The distance for on-policy primal representation $\mathbf{q}$ is measured by the norm defined in Equation 3.28; the distance for on-policy dual representation H is measured by the norm defined in Equation 3.29.

For the off-policy $\mathcal{M}$ operator, the plots show the distance from the current estimates to the optimal values (either $\mathbf{q}^*$ or H^*). The distance for off-policy primal representation $\mathbf{q}$ is measured by the max norm defined in Equation 3.47. The distance for off-policy dual representation H is measured by the norm defined in Equation 3.48.

[8] Tabular on-policy ($\mathcal{O}$), projection on-policy ($\mathcal{PO}$), gradient on-policy ($\mathcal{GO}$), tabular off-policy ($\mathcal{M}$), projection off-policy ($\mathcal{PM}$), and gradient off-policy ($\mathcal{GM}$) for both the primal and the dual.

Figures 3.2, 3.4, and 3.6 show the behavior of the on-policy update operators on different problems with both primal and dual representations. Figures 3.3, 3.5, and 3.7 show the behavior of the off-policy update operators on different problems with both primal and dual representations.

3.8.1 Task: Randomly Synthesized MDPs

For the synthesized MDPs, I generated the transition and reward function of the MDPs randomly—the transition function is uniformly distributed between 0 and 1 and the reward function is normally distributed with mean 0 and variance 1. Since my goal is to investigate the convergence of the algorithms without carefully crafting features, I also choose random basis functions according to standard normal distribution for primal representations, and random basis distributions according to uniform distribution for dual representations.

Here I only reported the plots of random MDPs with 100 states, 5 actions, and 10 bases, averaging over 100 repeats because I observed consistent convergence using dual representations with various MDP problems with different number of states, actions, and bases (the trends of the curves are similar).

In Figure 3.3, the curve (solid line with the circle marker) of the gradient off-policy ($\mathcal{GM}$) update with state-action value blows up (diverge) while all the other curves (algorithms) in Figures 3.2 and 3.3 converge. Interestingly, the approximate error of the algorithm $\mathcal{POH}$ (4.60×10^{-3}) is much smaller than the approximate error of the algorithm $\mathcal{PO}\mathbf{q}$ (4.23×10^{-2}) although their theoretical bounds are the same (see Figure 3.2).

3.8.2 Task: The Star Problem

The star problem has 7 states and 2 actions as Figure 3.1 shows. The reward function is zero for each transitions. The transitions in solid lines are triggered by action a_1 and the transitions in dotted lines are generated by action a_2, that is, taking action a_1 in all the states will cause a transition to state s_7 with probability 1; taking action a_2 in each state will cause a transition to one of states s_1 through s_6 with equal probability $1/6$.

In my experiments, I used the same fix policy and linear value function approximation as Baird did.[9] The fixed policy chooses action a_1 with probability $1/7$ and action a_2 with probability $6/7$. How the action values are represented can be figured out from Figure 3.1. The number of bases is 14 and the dimension of the weight vector is 14×1. The i^{th} component of the weight vector is w_i where $i = 0, \cdots, 13$. The action values are given by the linear combination of the weights. For example, $Q(s_1, a_1) = w_0 + 2w_1$ and $Q(s_1, a_2) = w_7$. The initial action values of a_1 are bigger than the ones of a_2 and the value of the action a_1 in state six is the largest action values. In the dual, the number of bases is 14 too and the initial values of the state-action visit distribution matrix H are uniformly distributed random numbers between 0 and 1 with row normalization.

The gradient off-policy update in the primal case diverges (see the solid line with the circle marker in Figure 3.5). However, all the updates with the dual representation algorithms converge (see Figures 3.4 and 3.5).

3.8.3 Task: The Mountain Car Problem

The mountain car domain has continuous state and action spaces, which I discretize with a simple grid, resulting in an MDP with 222 states and 3 actions. The number of bases is chosen to be 5 in both the primal and dual algorithms. For the same reason as before, I chose the bases for the algorithms randomly. In the primal representations with linear function approximation, I randomly generated basis functions according to the standard normal distribution. In the dual representations, I randomly picked the basis distributions according to the uniform distribution.

In Figure 3.7, I again observed blow-up of the gradient off-policy update with state-action value in the primal and the convergence of all the dual algorithms (see Figures 3.6 and 3.7). Interestingly, the approximation error of the projection on-policy $\mathcal{P}OH$ in the dual (0.19) is also considerably smaller than the one $\mathcal{P}Oq$ (3.26×10^2) in the primal.

In summary, the dual dynamic programming algorithms in fact converge in the

[9] Baird observed that Q-learning with linear function approximation can diverge on this problem even when training on a fixed stochastic policy (Baird, 1995).

Update	Primal Representation q	Dual Representation H
$\mathcal{O}$	$O(mn)$	$O(m^2n)$
$\mathcal{M}$	$O(m)$	$O(m)$
$\mathcal{P}$	$O(mk + k^3)$	$O(m^2k + k^3)$
$\mathcal{G}$	$O(mk)$	$O(m^2k)$
$\mathcal{PO}$	$O(mn + mk + k^3)$	$O(m^2n + m^2k + k^3)$
$\mathcal{GO}$	$O(mn + mk)$	$O(m^2n + m^2k)$
$\mathcal{PM}$	$O(mk + k^3)$	$O(m^2k + k^3)$
$\mathcal{GM}$	$O(mk)$	$O(m^2k)$

Table 3.1: Computational complexity of the dynamic programming algorithms: tabular on-policy ($\mathcal{O}$), projection on-policy ($\mathcal{PO}$), gradient on-policy ($\mathcal{GO}$), tabular off-policy ($\mathcal{M}$), projection off-policy ($\mathcal{PM}$), and gradient off-policy ($\mathcal{GM}$) updates for both the primal and the dual representations. The quantity n is the number of states, i.e. $n = |S|$. The quantity m is the product of the number of states and the number of actions, i.e. $m = |S||A|$. The quantity k is the number of bases (features) in linear approximation case. Usually, k is much less than $|S|$ and $|A|$.

very circumstance where primal algorithm can diverge: gradient off-policy update with linear function approximation. The experimental results of randomly synthesized MDPs and the mountain car problem showed that off-policy updates diverge when composed with gradient ($\mathcal{GM}$) in the primal with randomly picked basis functions while $\mathcal{GM}$ converges in the dual with randomly picked basis distributions.

3.9 Complexity Analysis of Dual Algorithms

Although dual representations are interesting, dual algorithms can be computationally much more expensive than the corresponding primal algorithms, ignoring possible sparsity of the dual representations. Table 3.1 is a summary of the complexity of the dynamic programming algorithms presented in this chapter. The cost of the composed update operators are the sum of the costs of its operators. For example, the cost of $\mathcal{PO}$ is the sum of cost of $\mathcal{O}$ and cost of $\mathcal{P}$.

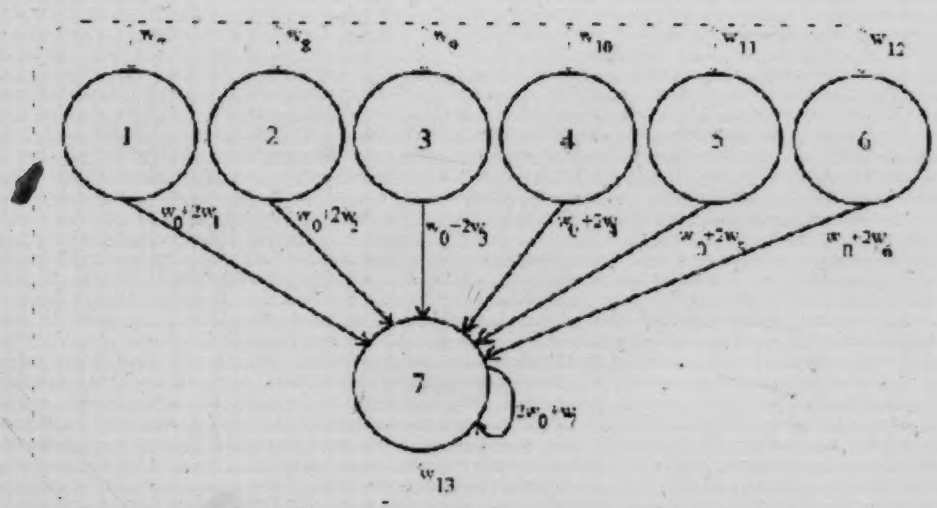

Figure 3.1: The star problem

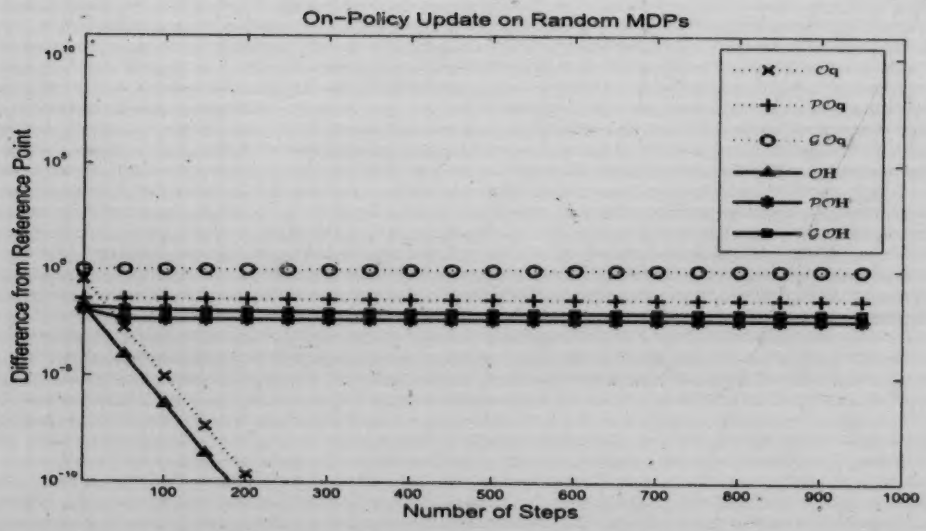

Figure 3.2: On-policy update of state-action value q and visit distribution H on randomly synthesized MDPs

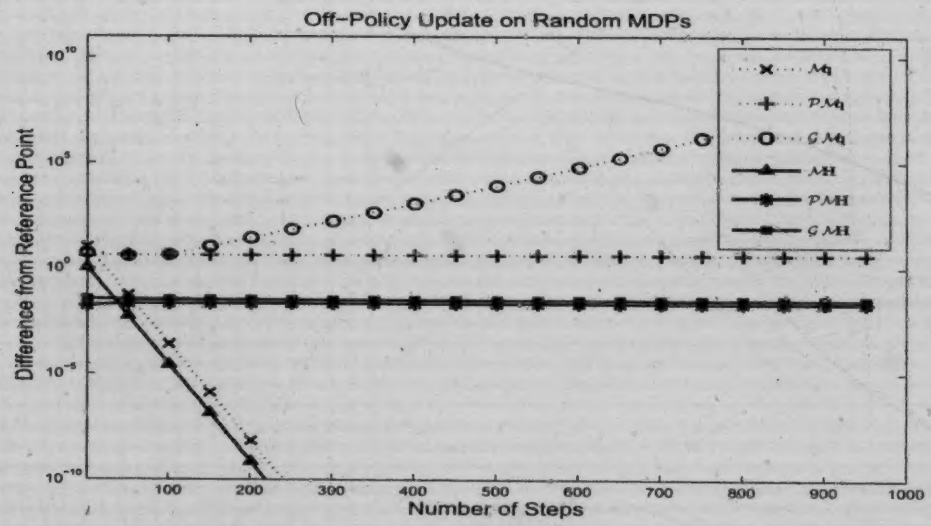

Figure 3.3: Off-policy update of state-action value q and visit distribution H on randomly synthesized MDPs

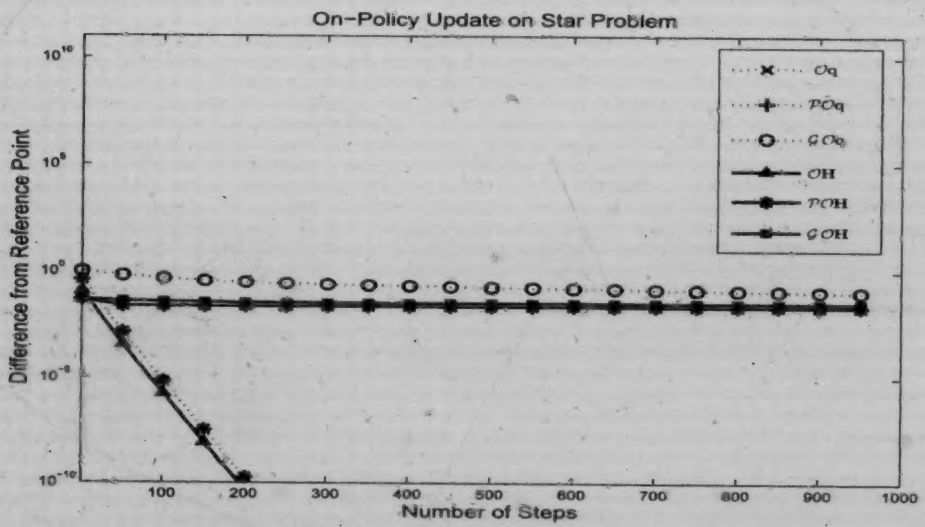

Figure 3.4: On-policy update of state-action value q and visit distribution H on the star problem

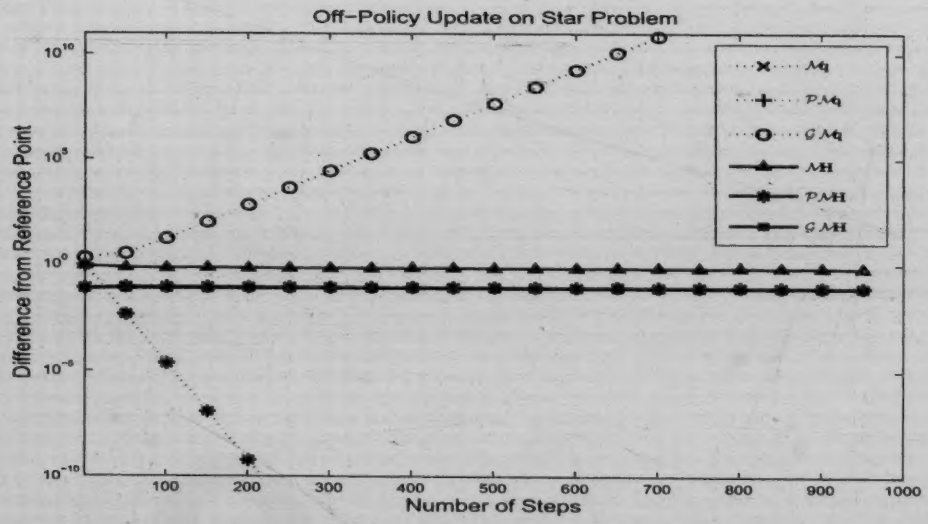

Figure 3.5: Off-policy update of state-action value q and visit distribution H on the star problem

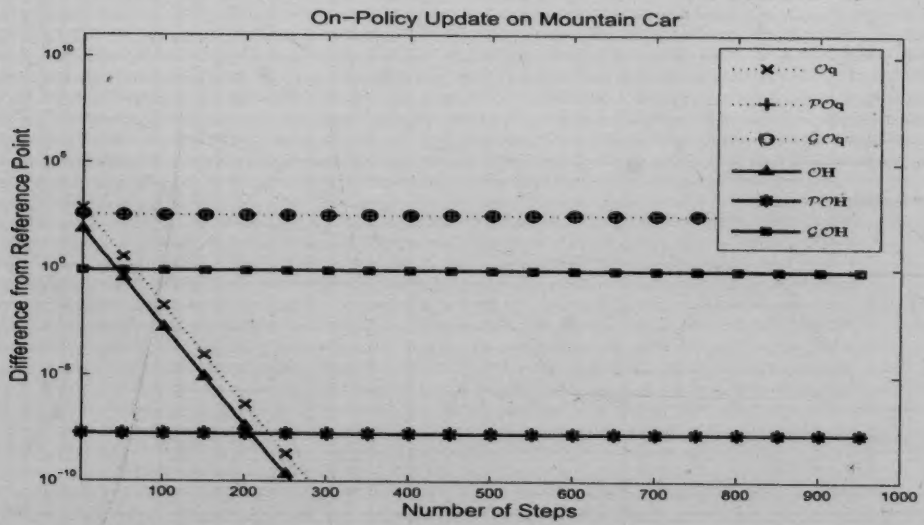

Figure 3.6: On-policy update of state-action value q and visit distribution H on the mountain car problem

3.10 Conclusion

I have introduced dual representations and a body of novel dual algorithms for sequential decision making problems. Dual representations maintain an explicit representation of visit distributions as opposed to value functions. Therefore, dual algorithms, since they are based on estimating normalized probability distributions rather than unbounded value functions, avoid divergence even in the presence of approximation and off-policy updates. Moreover, dual algorithms remain stable in situations where standard value function estimation diverges. One limitation of the dual approach is that the updates in the dual algorithms are more expensive than the ones in the primal case. However, this limitation may be tackled by exploiting the sparsity in the dual matrix representations, which might be the case for some decision making problems.

I studied the convergence properties of the dual dynamic programming algorithms both theoretically and empirically. I observed that on-policy updates converge in both primal and dual algorithms while off-policy updates diverge when composed with gradient operator in the primal, not in the dual. Learning from the techniques for proving convergence for the primal algorithms, I proved the convergence of tabular on-policy ($\mathcal{O}$), tabular off-policy ($\mathcal{M}$), and projection on-policy ($\mathcal{PO}$) updates for the dual. The reason is that these updates are non-expansion (or contraction) operators with respect to the right norm. The experimental studies show the convergence of the projection off-policy ($\mathcal{PM}$), gradient on-policy ($\mathcal{GO}$), and gradient off-policy ($\mathcal{GM}$) in the dual, however these were not proved.

In the next chapter, I will present my attempt to the second issue: how can we balance the fundamental trade-off between exploration and exploitation during action selection?

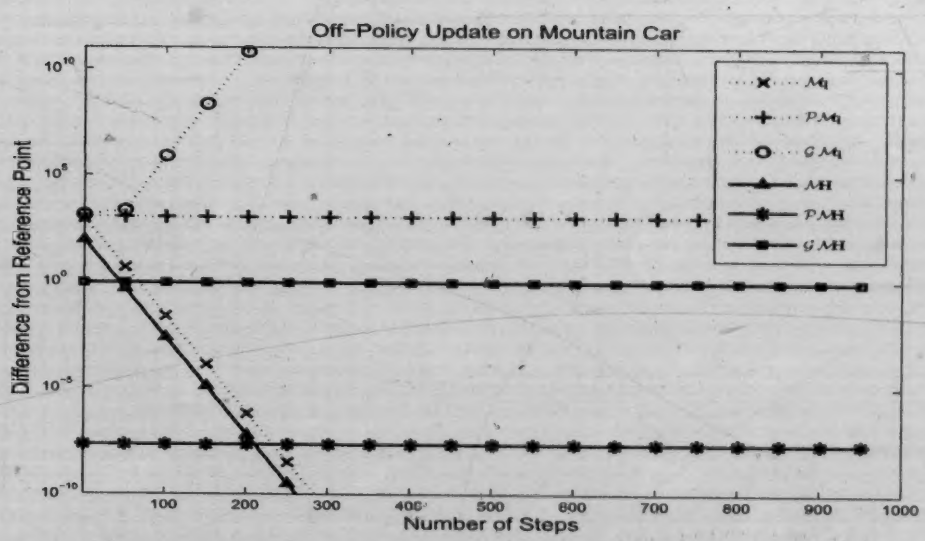

Figure 3.7: Off-policy update of state-action value q and visit distribution H on the mountain car problem

Chapter 4

A Sparse Sampling Approach to Action Selection

In this chapter, I address the well-known problem of balancing exploitation with exploration, or more generally, the problem of action selection during reinforcement learning from a Bayesian point of view. My approach combines sparse sampling with Bayesian exploration to achieve improved decision making, while controlling computational cost. The idea is to exploit a Bayesian posterior to make intelligent action selection decisions by constructing and searching a sparse lookahead tree. The outcome is a flexible and relatively straightforward technique for improving action selection in simple reinforcement learning scenarios (Wang et al., 2005). The limitation of this work is that it was only demonstrated on the multi-armed bandit problems where the planning (for computing the values of the leaf nodes in the sparse lookahead tree) is relatively easy.

This chapter is organized as follows. Section 4.1 motivates the problem. Section 4.2 reviews the basics in Bayesian Statistics. Section 4.3 briefly surveys Bayesian reinforcement learning. Section 4.4 discusses existing action selection strategies. Section 4.4 then presents my idea of Bayesian sparse sampling for on-line action section. Section 4.6 demonstrates that my idea improved action selection quality in simple reinforcement learning domains. Finally, Section 4.7 summarizes the contributions of this work.

4.1 Motivation

Action selection is fundamental in reinforcement learning. Although many strategies (Kaelbling, 1994; Dearden et al., 1999; Strens, 2000; Wyatt, 2001; Strehl & Littman, 2005) have been proposed to address this issue, few techniques have been adopted beyond the papers that originally proposed them other than the standard ϵ-greedy and Boltzmann selection strategies. A possible reason for the limited use of sophisticated exploration approaches might be the complexity of implementing the proposed methods, or the assumption that the degree of improvement might not be dramatic. Therefore, beyond the quality of action selection results, it is also important to consider the complexity and computational cost of any proposed method.

The Bayesian approach to reinforcement learning appears to be under-researched given the important role it has played in other areas of machine learning (Jordan, 1999; Neal, 1996). Flexible Bayesian tools, such as Gaussian process regression (Williams, 1999; Neal, 1996), have had a significant impact on other areas of machine learning research but have only just recently been introduced to reinforcement learning (Engel et al., 2003). Nevertheless, Bayesian approaches seem suited to reinforcement learning as they offer an explicit representation of uncertainty, which is essential for reasoning about the exploration versus exploitation tradeoff. Bayesian decision theory suggests to solve the exploration versus exploitation tradeoff directly (but implicitly) by asserting that the optimal action is one which, over the entire time horizon being considered, maximizes the total expected reward averaged over possible world models. Therefore, any gain in reducing uncertainty is not valued for its own sake, but measured instead in terms of the gain in future reward it offers. In this way, explicit reasoning about exploration versus exploitation is subsumed by direct reasoning about rewards obtained over the long term.

Despite the elegance of the Bayesian approach, there remain serious barriers to its application. The most obvious drawback is the computational challenge posed by optimal Bayesian decision making, which is known to be intractable in all but trivial decision making contexts (Mundhenk et al., 2000; Lusena et al., 2001). This means that with a Bayesian approach one is forced to consider heuristic approxi-

mations. In response, a small body of research has developed on on-line approximations of optimal Bayesian action selection (Dearden et al., 1999; Duff, 2002; Strens, 2000). However, the potential power of Bayesian modeling for approximating optimal action selection makes this approach worth investigating.

Before I present my idea for on-line action selection in Bayesian reinforcement learning. First, I review the basic concepts of Bayśian statistics.

4.2 Bayesian Statistics

Bayesian statistics provide a rationalist theory of subjective beliefs in the context of uncertainty. In particular, Bayes' rule provides the key to combining beliefs in the light of new evidence, such as observed data. In the following sections, I will review some important concepts in Bayesian statistics and more information can be find in text (Bernardo & Smith, 2000).

The Subjective View of Probability. Probability can be viewed as either a relative frequency of the observation of an outcome during many repeatable experiments (*objective* or *frequentist*) or an individual's personal assessment of an outcome's likelihood (*subjective* or *Bayesian*). Throughout this chapter, I adopt a subjective interpretation of probability. Let $p(X)$ denote the probability of a random variable or vector X: p denotes a probability density function and $\int$ denotes an integration if X is continuous; otherwise if X is discrete, P denotes a probability mass function and $\int$ denotes a summation. $p(x)$ measures the agent's level of confidence or "degree of belief" in the likelihood of X taking value x.

Bayes' Rule. Let a random variable or vector $\theta \in \Theta$ denote the unknown parameter of a model, where Θ characterizes a class of possible models for a given problem, and let x represent the observed data. The prior probability of the model parameter $p(\theta)$ reflects our prior knowledge about the given problem. The likelihood function $p(x \mid \theta)$ reflects the conditional probability of observing data x given a model that is specified by parameter θ. Bayes' rule allows us to compute the

posterior probability of model parameter θ after observing the data x as follows:

$$p(\theta \mid x) = \frac{p(x \mid \theta)p(\theta)}{p(x)} \tag{4.1}$$

where

$$p(x) = \int_{\theta \in \Theta} p(x \mid \theta)p(\theta)d\theta \tag{4.2}$$

Equation 4.1 can be explained informally as follows:

$$posterior = \frac{likelihood \times prior}{normalization\ factor} \tag{4.3}$$

That is, a posterior distribution can be calculated by multiplying a prior distribution with a likelihood function and then normalizing.

In Bayesian statistics, Bayes' rule inverts the statistical connections, and shows how the probability distribution of θ changes from the prior probability $p(\theta)$ to the posterior probability $p(\theta \mid x)$ after we have observed the data x. Bayes' rule provides a formal framework for uncertainty analysis and decision making as well as a solution to the problem of how to learn from data.

Conjugate Priors. In general, the calculation of the probability in Equation 4.2 (normalization factor) can be very difficult. However, for certain choices of a prior, a difficult numerical integration can be avoided if a posterior has the same functional form as its prior (i.e., the prior distribution and the posterior distribution are in the same family of distributions). Such a choice is called a *conjugate prior*, and a prior and posterior chosen in this way are said to be *conjugate*.

Conjugate priors make Bayesian estimation procedures very straightforward because they can be simply expressed in terms of using the sufficient statistics of the observed values to update the parameters of the conjugate prior. Some useful conjugate priors can be found in Appendix E.

The choice of the functional form of the prior (posterior) depends on the likelihood, which is determined by the nature of the data-generating process. For example, if one is estimating the unknown parameters that define the probability distribution of a Binomial random variable, then a Beta distribution is a common choice of prior for those parameters. Since the Beta distribution is conjugate, the posterior

is another Beta distribution. Similarly, if one is estimating the mean parameter of a Gaussian distribution, the use of a Gaussian prior on the mean will lead to another Gaussian posterior. I will make particular use of Gaussian processes below, which are a generalization of multivariate Gaussian to general index sets. Appendix D reviews several specific probability models that I exploit below.

So far I have covered basic concepts in Bayesian statistics. In the next section, I will give a brief survey of Bayesian reinforcement learning.

4.3 Bayesian Reinforcement Learning

The literature on *Bayesian* reinforcement learning is relatively small, although Bayesian approaches were considered almost from the beginning (Martin, 1967; Bellman, 1961). However, interest has re-emerged in this approach (Engel et al., 2003; Dearden et al., 1999; Strens, 2000; Wyatt, 2001). In the Bayesian approach to reinforcement learning, *uncertainty* is represented explicitly as a probability distribution over an MDP model (*model based*) or the (action) value function (*value based*). In this way, the estimates of the unknown parameters of the model (or the value function) take into account the fact that the estimates themselves are uncertain.

Overall, Bayesian modeling is a flexible tool that allows prior knowledge about the transition and reward models to be explicitly stated. It also allows generalization across actions, states and rewards, through a principled mechanism. Some of the best developed Bayesian modeling tools, such as Gaussian processes (Williams, 1999), are suited specifically for continuous state and action spaces, where classical reinforcement learning methods are not always conveniently applicable. Bayesian approaches also naturally provide an explicit representation of *uncertainty* in the posterior distribution, which is useful for exploration/exploitation decision making. I attempt to exploit all of these advantages in Section 4.5.

Model-based Bayesian Exploration

The most straightforward Bayesian approaches are generally *model based*. In this case, a prior is kept over the underlying MDP (i.e., the transition and reward models, $\mathcal{T}$ and $\mathcal{R}$) and learning consists essentially of updating the posterior.

The model based Bayesian approach to reinforcement learning, as I review, is equivalent to choosing actions in a meta-level MDP. This meta-level problem is sometimes referred to as a belief state MDP or a Bayes-adaptive MDP (Duff, 2002). I will refer to the underlying MDP as the base-level MDP.

Let θ and ξ to denote the unknown parameters of the state transition and reward models of the underlying MDP, respectively. Given a particular θ and ξ, the corresponding base-level MDP is given by the tuple $\langle S, A, \mathcal{T}^\theta, \mathcal{R}^\xi \rangle$. However, the parameters θ and ξ are not precisely known, but instead assumed only to belong to a general set, $\theta \in \Theta$ and $\xi \in \Xi$. We assume in particular that a *prior* distribution $P(\theta, \xi) = P(\theta, \xi | s_0)$ is defined over the models of the base-level MDP. The prior is typically assumed to be factored into separate transition and reward models $P(\theta, \xi | s_0) = P(\theta | s_0) P(\xi | s_0) = p_0^\theta(\theta) p_0^\xi(\xi)$. Let $p_t^\theta \triangleq P(\theta | s_0 a_0 ... s_{t-1} a_{t-1} s_t)$ and $p_t^\xi \triangleq P(\xi | s_0 a_0 r_1 ... s_{t-1} a_{t-1} r_t)$. Given experience $s_0 a_0 r_1 s_1 ... s_{t-1} a_{t-1} r_t s_t$, one can calculate the *posterior* distribution $P(\theta, \xi \mid s_0 a_0 r_1 s_1 ... s_{t-1} a_{t-1} r_t s_t) = p_t^\theta p_t^\xi$ using Bayes' rule.

The decision maker begins in an initial meta-level MDP state $m_0 = \langle s_0, p_0^\theta, p_0^\xi \rangle \in \mathcal{M}$. At each time step t, it executes an action $a_t \in \mathcal{A}$, and then receives a reward r_t drawn from the reward distribution specified by $\mathcal{R}(m_t, a_t, r_t)$ (given in Equation 4.5 below), and finds itself in a new meta-level state $m_{t+1} = \langle s_{t+1}, p_{t+1}^\theta, p_{t+1}^\xi \rangle$, where $s_{t+1} \in \mathcal{S}$. In fact, the meta-level states $m_t = \langle s_t, p_t^\theta, p_t^\xi \rangle$ are equivalent to histories $m_t \equiv s_0 a_0 r_1 ... s_{t-1} a_{t-1} r_t s_t$, and the state transition probability is simply the probability of a particular history extension r_{t+1}, s_{t+1} given the current history $s_0 a_0 r_1 ... s_{t-1} a_{t-1} r_t s_t$ and action a_t.

Formally, the meta-level Markov decision process can be defined by a tuple $\langle \mathcal{M}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where

- $\mathcal{M}$ is the set of meta-level states $\mathcal{M} = \mathcal{S} \times \{P(\theta)\} \times \{P(\xi)\}$,
- A is the set of actions,
- $\mathcal{T}$ is a transition (probability) function, $\mathcal{T}(m, a, m') = P(m' \mid m, a)$,
- $\mathcal{R}$ is a reward (probability) function, $\mathcal{R}(m, a, r) = P(r \mid m, a)$,

where $m \in \mathcal{M}$, $m' \in \mathcal{M}$, $a \in A$, $r \in \mathbb{R}$. The meta-level transition model is given by

$$\begin{aligned}\mathcal{T}(m_t, a_t, m_{t+1}) &= P(\langle s_{t+1}, p^{\theta}_{t+1}, p^{\xi}_{t+1}\rangle \mid \langle s_t, p^{\theta}_t, p^{\xi}_t\rangle, a_t) \\ &= 1_{[p^{\theta}_{t+1} = P(\theta \mid s_0 a_0 \ldots s_{t+1})]} \left[\int_{\theta \in \Theta} \mathcal{T}^{\theta}(s_t, a_t, s_{t+1}) p^{\theta}_t(\theta) d\theta\right] \\ &\quad \int_{r_t \in \mathbb{R}} 1_{[p^{\xi}_{t+1} = P(\xi \mid s_0 \ldots s_t a_t r_{t+1})]} \int_{\xi \in \Xi} \mathcal{R}^{\xi}(s_t, a_t, r_{t+1}) p^{\xi}_t(\xi) d\xi dr_{t+1}\end{aligned} \tag{4.4}$$

and the meta-level reward model is then simply given by the expectation

$$\mathcal{R}(m_t, a_t, r_{t+1}) = \int_{\xi \in \Xi} \mathcal{R}^{\xi}(s_t, a_t, r_{t+1}) p^{\xi}_t(\xi) d\xi \tag{4.5}$$

Interestingly, even though the base-level transition and reward models are not known, the meta-level transition and reward models are known, and hence define a known meta-level MDP.

Reinforcement Learning with Gaussian Processes

The above discussion concerns a *model based* approach to Bayesian reinforcement learning. Recently, there has been an attempt to pursue a more direct, value based approach to Bayesian reinforcement learning.

The idea is to use Gaussian processes as a tool for estimating the action value function of a fixed policy (unfortunately not the optimal value function) from a Bayesian perspective (Engel et al., 2003; Rasmussen & Kuss, 2004). Engel and his colleagues put a Gaussian prior over the value function of a fixed policy and assume that the reward distribution is also Gaussian. Then the Bellman equation update for the policy's value function (see Equation 2.7) can be maintained by a Gaussian process posterior update as shown in Section F.1.

Although this is an elegant approach, it is difficult to maintain the Gaussian process representation when introducing the *max* operator for the optimal value function Bellman's optimality principles (see Equations 2.8 and 2.12). Although some work (Engel et al., 2003; Dearden et al., 1998) has considered a Bayesian

approach to value based learning, I consider a model based approach to Bayesian reinforcement learning in my work.

4.4 Action Selection

Sections 4.2 and 4.3 presented the general technical background about Bayesian statistics and Bayesian reinforcement learning. However, beyond estimating models, value functions, or policies, another key question in learning is *action selection*. That is, which actions should be executed during learning to ensure that a large value of reward is obtained over the agent's lifetime or within a learning episode? This section reviews the existing techniques for addressing the problem of action selection.

Previous research on action selection can be classified according to whether the reward accumulated by the agent during the learning matters or not. The distinction is captured in the terms *on-line* and *off-line* reinforcement learning problems. On-line learning is a very natural model for reinforcement learning as it captures the characteristics of a learner that is improving its action selection while performing actions. Off-line or batch learning distinguishes an initial training phase from a subsequent testing phase. During training, the learning algorithm has no responsibility to obtain reward and focuses solely on gaining information. During subsequent testing, a non-adaptive policy is executed. The goal of the agent is actually very different between on-line learning and off-line learning. The goal of an on-line learning agent is to maximize the total reward it can accumulate during learning. However, the goal of an off-line agent is to reduce the time spent learning; that is, to minimize the total number of actions executed to achieve a policy that is approximately optimal, which is not the same as maximizing the total reward obtained along the way. I am much more interested in on-line learning so I will only briefly mention some work on batch learning in Appendix F.3.

Here existing on-line action selection strategies (see Table 4.1) will be categorized according to their representation of the uncertainty of their estimates (e.g., estimates of action values). If uncertainty is represented as a prior (posterior) distri-

bution, the strategies are considered as Bayesian approaches; otherwise, they belong to the non-Bayesian approaches. Non-Bayesian strategies, which are all myopic and do not consider the future estimates of the action value, are first introduced. Then the Bayesian formulation of action selection is introduced.

Method	Bayesian	Lookahead	Uncertainty
ϵ-greedy	—	—	—
Boltzmann	—	—	—
Interval Estimation	—	—	Confidence Interval
Bayes Optimal	✓	✓	Probability Distr.
Value of Perfect Information	✓	—	Probability Distr.
Thompson Sampling	✓	—	Probability Distr.

Table 4.1: On-line methods for action selection

4.4.1 Non-Bayesian Methods for Action Selection

First I will review the most commonly used non-Bayesian action selection strategies: ϵ-greedy, Boltzmann, and interval estimation. Non-Bayesian approaches very often assume that a point estimate of the action value function, $\hat{Q}(s,a)$, is maintained (either by Q-learning or Sarsa or some other means). Based on these point estimates, some decision needs to be made about which action to take in a current state s.

ϵ-greedy. ϵ-greedy is the most commonly used strategy to balance exploration and exploitation in reinforcement learning. The idea is to choose the action with the highest estimated action value, that is $a^* = \arg\max_a \hat{Q}(s,a)$, with probability $1-\epsilon$, or choose a (uniform) random action $a \in A$ with probability ϵ.[1] It is also referred to as semi-uniform random exploration. It is common to decrease ϵ over the learning process. ϵ-greedy is popular for its simplicity and it is widely used in reinforcement learning algorithms (Sutton & Barto, 1998). However, it ignores the estimated action values of the non-greedy actions, and instead treats them all equally. As a consequence, ϵ-greedy will select promising actions with the same

[1]For infinite action spaces we assume the range of possible actions is bounded.

probability as ones that are known to be poor. A more sophisticated action selection strategy, Boltzmann exploration, takes these differences into account.

Boltzmann Exploration. The Boltzmann exploration strategy (Luce, 1959), also called the softmax strategy, randomly selects actions according to their estimated action values, so that the probability of an action being selected increases with the current estimate of its action value. That is, actions with high estimated values (relatively good actions) are more likely to be chosen while actions with low estimated values (relatively poor actions) are less likely to be taken. Formally, the Boltzmann strategy samples a random action according to the probability $P(a|s) = \exp\left(\hat{Q}(s,a)/\tau\right)/Z$ where τ is a temperature parameter and Z is a normalization constant. When the temperature parameter $\tau \to \infty$, the Boltzmann strategy behaves like uniform random action selection; when the temperature parameter $\tau \to 0$, the Boltzmann exploration behaves like greedy action selection. However, Boltzmann exploration does not account for the uncertainty of the action value estimates.

Interval Estimation. The basic intuition behind interval estimation (Kaelbling, 1994) is that the greater the uncertainty in an action's value, the greater the chance that it might actually prove to be optimal, and therefore we should select it with a greater probability. Thus, considering the uncertainty of the estimated action value, interval estimation adds the confidence level to the estimated action value and chooses an action according to $a = \arg\max_a[\hat{Q}(s,a) + U(s,a)]$ where $U(s,a)$ is a $(1-\delta)$ upper bound of the confidence interval on the point estimate $\hat{Q}(s,a)$.[2]

Although generally interval estimation is thought to be more sophisticated and more effective than ϵ-greedy or Boltzmann exploration, it is still a myopic strategy. That is, the above action selection strategies (ϵ-greedy, Boltzmann, and interval estimation) do not explicitly consider the effects that actions have on future value estimates. Interval estimation tries to overcome its myopia by using uncertainty as a proxy for lookahead. One difficulty with this type of intuitive reasoning, however,

[2]This approach has been extended to general MDPs (Wiering, 1999; Strehl & Littman, 2005) and bandit-based Monte-Carlo planning (Kocsis & Szepesvári, 2006).

is that it is hard to quantify and ignores horizon effects (i.e., one would like to exploit more heavily if nearing the end of a decision making process). These three selection procedures are heuristic, sometimes difficult to justify, and do not perform well in all circumstances, as we will see below.

There are a few other simple non-Bayesian action selection strategies that people have considered, particularly for bandit problems.

Uniform Random. This strategy chooses actions randomly and uniformly. Every action has an equal probability of being selected.

Round Robin. This method selects actions one after another according to some fixed order. This is a potentially useful strategy when trying to gather data about all the actions.

Biased Robin. Given that the actions are arranged in some fixed order, the biased Robin method will stick to an action as long as it receives a reward that is higher than some fixed value; otherwise, it switches to the next action with wrap-around.

Probability Matching. Given pointwise estimates for the value of each action, the probability matching method normalizes each estimate by dividing it over the sum of the estimates, then selects an action according to the normalized probability.

4.4.2 Bayesian Action Selection

Classically, action selection in reinforcement learning has not been thought of in Bayesian terms but instead tackled intuitively. Typically, given a current Q-function estimate $\hat{Q}(s, a)$, an exploitation step is one where an optimal or near optimal action is selected according to the current estimate, whereas an exploration step involves choosing an alternative action, hoping that its current low estimated value is just an underestimate.

Bayes Action Selection. A conceptually elegant solution to the action selection problem is offered by Bayesian decision theory. A Bayesian approach to learn-

ing optimally in a Markov decision process is equivalent to solving for an optimal action selection strategy in the meta-level Markov decision process (see Equations 4.4 and 4.5). That is, an optimal action selection strategy for reinforcement learning in this setting is given by the policy that obtains maximum expected reward in the meta-level Markov decision process. However, even though this formalization characterizes optimal action selection for Bayesian reinforcement learning, there is no efficient way to compute this strategy in a guaranteed way (Mundhenk et al., 2000; Lusena et al., 2001). One obvious difficulty is that there are far more meta-level states (i.e., base-level histories) than original base-level states. In all but trivial circumstances, there is no hope of exactly following an optimal action selection strategy. Perhaps the only well known exception to this is the result of Gittins (1989) which shows that in the special case where there are finitely many actions, each with their own independent (finite) state spaces (i.e., bandit problems[3]), then optimal action decisions can be made in polynomial time to maximize the expected sum of infinite horizon discounted rewards (see Appendix F.2). However, the restrictiveness of the independence assumption has prevented this approach from being widely applied in reinforcement learning problems. Beyond the work of Salganicoff & Ungar (1995) and Duff (2002), very few successes have been reported in this direction.

For the most part, work on approximating Bayes optimal action selection has followed two approaches: pre-compilation and on-line computation. In the *pre-compilation* approach, one attempts to derive a compact approximation to the optimal value function (as in Section 2.3.1) for the meta-level state MDP. In fact, any approximation strategy for general POMDPs is applicable in this case, although a few interesting specializations have been attempted for belief state MDPs (Duff, 2002). One potential shortcoming of the pre-compilation approach is that once an action selection strategy has been fixed, it is hard to adapt it to the meta-level states that are actually encountered during learning. Moreover, this approach necessarily cannot obtain a uniformly accurate approximation over the entire state and action

[3]The multi-armed bandit problem is a mathematical model for the problem of optimizing the total reward accumulated by choosing actions from a number of arms (e.g., a slot machine), whose reward distributions are unknown. An informal review of the multi-armed bandit problem is presented in Appendix F.2.

space, and there is no guarantee that the approximation holds over the meta-level states that are encountered in a particular learning episode. Pre-compilation might nevertheless be the only viable approach if actions need to be selected with little or no serious computation.

In contrast, the *on-line* approach to approximating Bayes optimal action selection attends only to the particular meta-level states encountered during learning, which would seem to relax the burden on the approximation strategy and offer the prospect of higher quality decisions. The drawback is that instead of extensive pre-compilation (allowing fast on-line decisions), these techniques can require non-trivial computation for each action selection decision. To date, on-line approximation strategies to Bayesian optimal decision making have all been myopic.

Thompson Sampling. One of the most interesting myopic action selection strategies in the Bayesian setting is in fact one of the first action selection strategies to have ever been proposed (Thompson, 1933; Thompson, 1935; Berry & Fristedt, 1985). In the context of learning in MDPs, Thompson sampling performs the following procedure. Given a current meta-level state $\langle s_t, p_t^\theta, p_t^\xi \rangle$, sample a transition and reward model, θ and ξ, from the distributions p_t^θ and p_t^ξ. Solve for the optimal action value function $Q_{\theta\xi}(s, a)$ for this model, then select the optimal action

$$a_t = \operatorname*{argmax}_{a \in A} Q_{\theta\xi}(s_t, a) \tag{4.6}$$

This technique was originally proposed by Thompson (1933) for bandit problems, and has recently reemerged in the reinforcement learning literature (Strens, 2000). Thompson sampling selects actions according to the *probability* that they are optimal in models drawn randomly from the current meta-level state. Although old, this remains an elegant and effective action selection strategy that often outperforms modern proposals (Berry & Fristedt, 1985). Thompson sampling is not Bayes optimal, however, as it is still myopic. In our experiments (see Section 4.6) we find that it tends to over-explore, which is obviously true at the horizon.

Like non-Bayesian myopic strategies, Thompson sampling does not explicitly account for the effects that actions have on future meta-level states, and therefore

can only supply proxy summaries for the future rewards that might indirectly accrue as the result of a current action.

Myopic Value of Perfect Information. A more recent action selection strategy is that of Dearden et al. (1998; 1999) and Wyatt (2001), which attempts to take the effects of exploration explicitly into account. This approach is based on considering the value that is gained by improving a Q-value estimate.[4]

Consider the distribution over action value functions, $Q_{\theta\xi}(s_t, a)$, defined by the current meta-level state $\langle s_t, p_t^\theta, p_t^\xi \rangle$, where $\theta \sim p_t^\theta$, $\xi \sim p_t^\xi$. Then, for each action $a \in \mathcal{A}$, consider the value of learning the exact value $Q^*(s_t, a)$ under the true model. Let a_1 and a_2 be the actions with the largest and second largest expected action values respectively. The *gain* in value of learning $Q^*(s_t, a)$ for an action a is given by definition

$$Gain\,(Q^*(s_t, a)) \;=\; \begin{cases} \left(\bar{Q}(s_t, a_2) - Q^*(s_t, a_1)\right)_+ & \text{if } a = a_1 \\ \left(Q^*(s_t, a) - \bar{Q}(s_t, a_1)\right)_+ & \text{otherwise} \end{cases} \tag{4.7}$$

where the mean Q-values $\bar{Q}(s, a)$ is defined as

$$\bar{Q}(s, a) \triangleq \mathrm{E}_{\theta \sim p_t^\theta, \xi \sim p_t^\xi}[Q_{\theta\xi}(s, a)] \tag{4.8}$$

are taken with respect to $\theta \sim p_t^\theta$, $\xi \sim p_t^\xi$.[5] (That is, value is gained only if a new action becomes the best, but not otherwise.) The value of learning the exact Q-value of an action in the current meta-level state is then simply given by the expected gain $VPI_t(a) = \mathrm{E}_{\theta\xi} Gain\,(Q_{\theta\xi}(s_t, a))$, which provides an upper bound on the myopic value of information of executing action a. Finally, one chooses the action that maximizes $\bar{Q}_t(s_t, a) + VPI_t(a)$.

Other Strategies. Many non-Bayesian strategies in fact can be reformulated in a Bayesian way. Bayesian variants of the ϵ-greedy, Boltzmann, and interval estimation action selection strategies are straightforward. One simply uses the *expected*

[4] Wyatt (1997) proposed Q-value sampling, which is also using Bayesian method for representing and propagating prior/posterior over action values. The idea was originally developed for solving bandit problems.

[5] Here we use the notation $(x)_+ = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$

action value function (see Equation 4.8) defined by the current meta-level state and select actions as described in Section 4.4.1.

The main drawback with these Bayesian action selection approaches is that computing the mean action value function, $\bar{Q}$, or even just computing the action value function for a sampled base-level MDP (as in Thompson Sampling) requires solving the planning problem in a base-level MDP. The fact that Bayesian on-line action selection strategies require (even limited) replanning for every meta-level state they encounter is probably the single greatest barrier to their use. Nevertheless, replanning is still viable in a range of interesting cases, which we will exploit in the Section 4.5. For example, planning is trivial in finite bandit problems and remains feasible in many episodic problems. Dearden et al. (1999) also show how importance sampling and prioritized sweeping can reduce the cost of replanning to just a few sampled models while maintaining reasonable estimates of $\bar{Q}_t(s, a)$.

In next subsection, I present my study on the problem of on-line action selection in relatively simple reinforcement learning problems (multi-armed bandits). The idea is to grow a sparse lookahead tree, intelligently, by exploiting information in a Bayesian posterior—rather than enumerate action branches (sparse sampling) or compensating myopically (value of perfect information). The outcome is a flexible, practical technique—"Bayesian sparse sampling"— for improving action selection in simple reinforcement learning scenarios.

4.5 Bayesian Sparse Sampling

My investigation on the problem of learning to behave optimally in an initially unknown MDP is a model based Bayesian approach discussed in Section 4.3. Recall that $\mathcal{T}^{\theta}(s_t, a_t, s_{t+1})$ and $\mathcal{R}^{\xi}(s_t, a_t, r_{t+1})$ denote the transition and reward models with unknown parameters θ and ξ respectively, and consider a learning scenario where the transition and reward parameters, θ and ξ, are not precisely known, but instead assumed only to belong to a general set, $\theta \in \Theta$ and $\xi \in \Xi$. Furthermore, assume we are given priors, $P(\theta)$ and $P(\xi)$, on θ and ξ respectively. As we have seen this creates a meta-level MDP whose state space, transition model and reward

model are now *known*. Optimal action selection in this meta-level MDP specifies the best possible action choices in the underlying Bayesian reinforcement learning problem. Unfortunately, it is not practical to solve the meta-level MDP, and the real question is how best to *approximate* Bayesian optimal action selection. Bayes optimal action selection essentially involves enumerating possible futures, averaging according to their realization probabilities, and choosing the best action. It is no surprise therefore that the only guaranteed way to approximate Bayes optimal action selection at a given meta-level state is to simulate the meta-level MDP to the effective horizon.

In my research, I exploit the sparse sampling technique of Section 2.3.2, which employs lookahead to derive a better approximation to Bayesian optimal action selection than myopic strategies. Note that sparse sampling requires a generative model, but this is conveniently exactly what a model-based Bayesian approach provides, as shown in Equations 4.4 and 4.5. In this approach, lookahead is performed only by simulation in the meta-level MDP which is maintained internally, not by actually taking actions in the world. That is, sparse sampling is an action selection strategy where, upon entering a meta-level state, extensive computation is performed to determine an action that would yield near optimal reward over the long run (i.e., to the horizon) in the meta-level MDP. Once chosen, the action is executed, and a new meta-level state is entered. To the extent that the Bayesian posterior concentrates on the true underlying model, this next meta-level state would have been influential in the previous computation. Surprisingly, sparse sampling has never been applied to approximating optimal action selection in Bayesian reinforcement learning before.

Even though sparse sampling can be parameterized to achieve a controllable lookahead strategy, it is still not an efficient action value estimator, and can be easily improved by addressing some shortcomings. My idea of addressing these shortcomings is to first grow a sparse lookahead tree adaptively, then evaluate the values of the nodes from leaves to the root by alternatively computing expectation at outcome nodes and maximum at the decision nodes.

First note that given a meta-level state $\langle s_t, p_t^\theta, p_t^\xi \rangle$, the goal is to use lookahead

search to estimate the long term value of possible actions. This situation is similar to game tree search where one wants to expand the lookahead tree (here an expectimax tree) intelligently so that search effort is not wasted and important branches are explored. One limitation of the sparse sampling algorithm in this respect is that it requires one to *enumerate* all possible actions to achieve its guarantee of approximating the optimal action with high probability. However, enumerating actions is not necessary in a Bayesian setting. The Bayesian approach has an advantage in that it allows one to approximate the maximum of a set of random variables without enumeration. Given a prior and sampled values, a posterior is determined over the distribution of the remaining variables. Thus, it is possible to stop whenever the expected posterior maximum value is no larger than the current maximum value, plus ϵ. In this way, it appears as though one can derive sparser sampling bounds in the Bayesian setting that are applicable to infinite action spaces. The main point is that it is not necessary to branch on every action to yield a good decision.

Grow a Tree. Thus my goal is not to build a fully balanced lookahead tree, but instead attempt to grow the tree adaptively. The intuition is that one need only investigate actions that are actually optimal, and not waste computational resources on proving that unpromising actions are, indeed, suboptimal. That is, uniformly accurate estimates are not required at every decision node in the lookahead tree. The next idea I pursued uses an effective myopic action selection strategy—specifically Thompson sampling—to preferentially expand the tree below actions that at least appear to be locally promising (see Figure 4.1). Also, to reduce the variance of the estimates at outcome nodes, I also exploit the fact that unbiased reward expectations, locally, can be obtained by sampling them from the mean model, rather than first sampling a model and then sampling rewards from a random model. Finally, although it seems subtle, the last idea I considered was to explicitly include the myopically *greedy* action at each decision node in the sparse lookahead tree. This small detail actually led to a significant improvement in action selection quality and is necessary to ensure an advantage over myopically greedy strategies. These ideas led to the algorithm shown in Table 4.2.

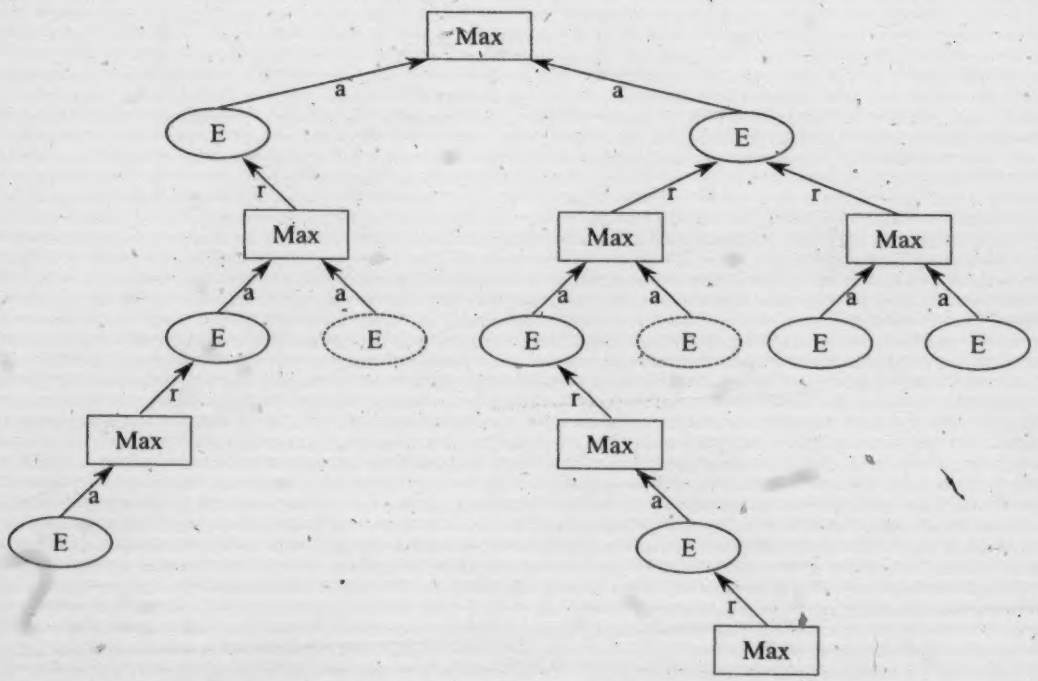

Figure 4.1: Illustration of an irregular lookahead tree that was generated by Bayesian sparse sampling idea.

GrowSparseBayesianTree (node, budget, ρ, horizon)

```
While # nodes < budget
    branchnode = BayesDescend(root, ρ)
    If branchnode.type = "decision"
      Add outcome then leaf node below branchnode
    If branchnode.type = "outcome"
      Add leaf decision node below branchnode
return EvaluateTree(root)
```

BayesDescend (node, ρ)

```
If node.type = "decision"
  a = ThompsonSample(node.belstate)
  If a ∉ node.children; return [node,a]
  Else return BayesDescend(node.child(a))
If node.type = "outcome"
  If possible to branch, with probability ρ; return node
  Else  [rew,obs] = sample(node.belstate,node.act)
      return BayesDescend(node.child([rew,obs]))
```

Table 4.2: Modified evaluation peocedure for sparse sampled trees that explicitly includes myopically.

Evaluate a Tree. Once grown, the sparse lookahead tree must be evaluated to choose an action at the root. There are a few subtleties in doing so effectively. Clearly, values are backed up from the leaves; averaging at outcome nodes, and maximizing at decision nodes, as shown in Figure 4.1. However, when evaluating leaf nodes (which are always decision nodes in this approach) it is important to account for differing depths. Therefore, at each leaf, the mean posterior reward for each action is first multiplied by the number of decisions remaining to the horizon, thus correcting the leaf values to the same absolute depth. Another important issue is to always consider the greedy action at each decision node, even if it was not sampled during the tree growing phase.[6]

Note that in this action selection procedure, myopic strategies are only used to decide where to look ahead in the simulation, not make any real action selection decisions. Real decisions are left to the full lookahead search. The procedure exploits the fact that there is much freedom, during lookahead, to make heuristic action choices at the internal decision nodes (i.e., max nodes). In fact, the Bayesian sparse sampling procedure can be easily applied to *infinite* action spaces, whereas sparse sampling is inapplicable if actions cannot be enumerated.

4.6 Experimental Results

To investigate the effectiveness of the improved sampling approach, I conducted experiments on a number of simple domains where the planning problem is not difficult. These include bandit problems, but also episodic reinforcement learning problems. My goal in this research is not to focus on MDP planning, but rather to demonstrate action selection improvements, which is already a challenge even in simple reinforcement learning scenarios. Subject to coping with possible MDP planning challenges (Dearden et al., 1999), the approach can be applied to a very rich class of domains.

I compared Bayesian sparse sampling (BayesSamp) with sparse sampling (Sparse-

[6] In the continuous action case I did not consider actions beyond those explicitly sampled, although additional local sampling could be used to ensure that a reasonable number of actions are considered at each decision node.

Samp) and standard myopic action selection strategies. These included Bayesian ϵ-greedy with $\epsilon = 0.1$ (eps-Greedy), Boltzmann exploration with temperature $\tau = 0.1$ (Boltzmann), and interval estimation (IE) with a range of two standard deviations, all using the expected Q-values given the current meta-level state. I also compared to Thompson sampling (Thompson) and the myopic value of perfect information (MVPI), using the same number of samples as a full lookahead tree of depth one to estimate the Q-value distributions. Finally, I compared to a lookahead strategy for action selection in MDPs proposed by (Péret & Garcia, 2004). For this strategy, independent trajectories to a fixed horizon H are generated (set to $H = 5$ in my experiments) and the action with the best overall trajectory reward on average is selected at the root.

For each problem domain, I set a finite horizon time T and measure the rewards accumulated by each action selection strategy, averaged over 1000 to $10,000$ repetitions to estimate the expected total reward achieved as a function of horizon time $T = 5, 10, 15, 20$. I also have collected the standard deviations for all these results which are reasonably small. The lookahead strategies were set up to give a controlled comparison with each other. First, sparse sampling was run with a given lookahead depth (1 or 2) and fixed decision and outcome branching factors, yielding a balanced tree. Then the total number of nodes expanded in the balanced tree generated by sparse sampling was set as a maximum node budget for both Bayesian sparse sampling and Péret & Garcia sampling. Figures 4.2 to 4.5 show the results obtained.

The first domain is a simple bandit problem with five actions, each yielding $\{0, 1\}$ rewards according to independent Bernoulli distributions with payoff probabilities distributed according to a Beta prior. Here one can see that lookahead strategies outperform the myopic strategies, even MVPI which uses comparable computation (Figure 4.2). Nevertheless this simple problem does not show much advantage for Bayesian over sparse sampling. Similar results were obtained for a related five action bandit problem where instead each action yields a reward according to an independent Gaussian distribution with means distributed according to a Gaussian prior (Figure 4.3).

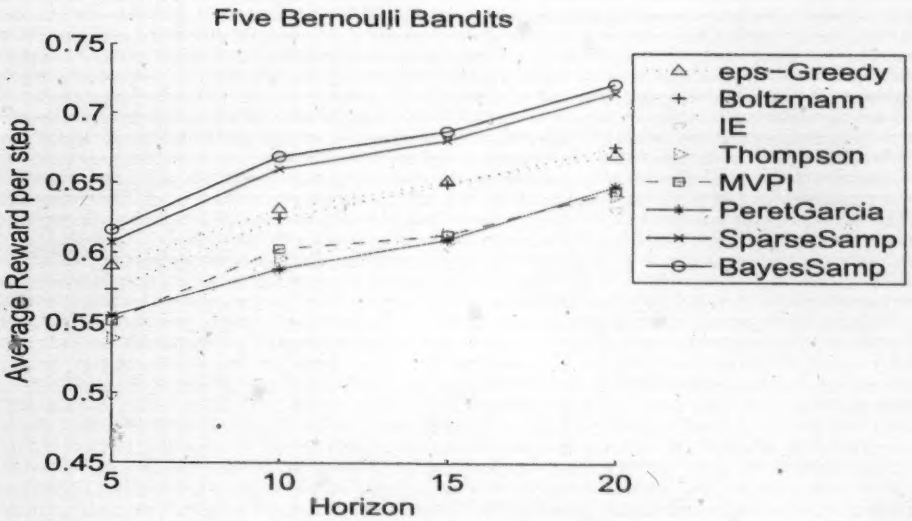

Figure 4.2: Results: Bernoulli bandits

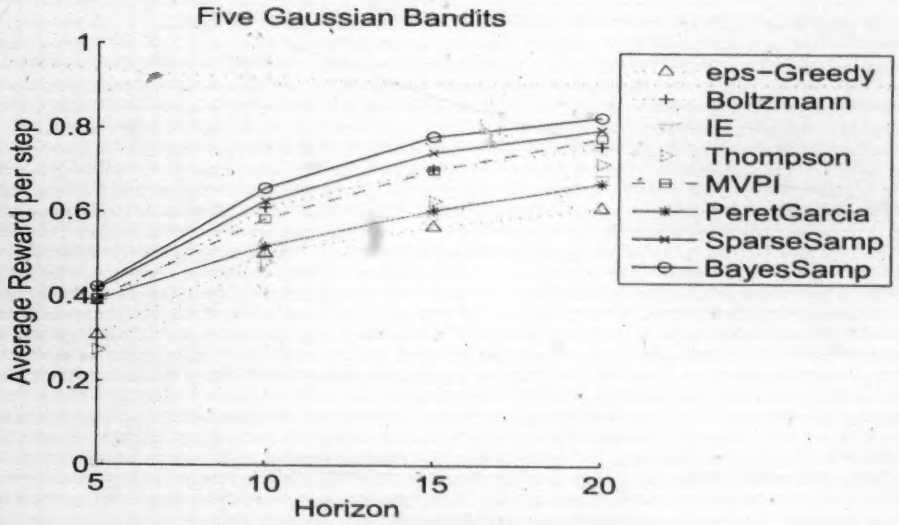

Figure 4.3: Results: Gaussian bandits

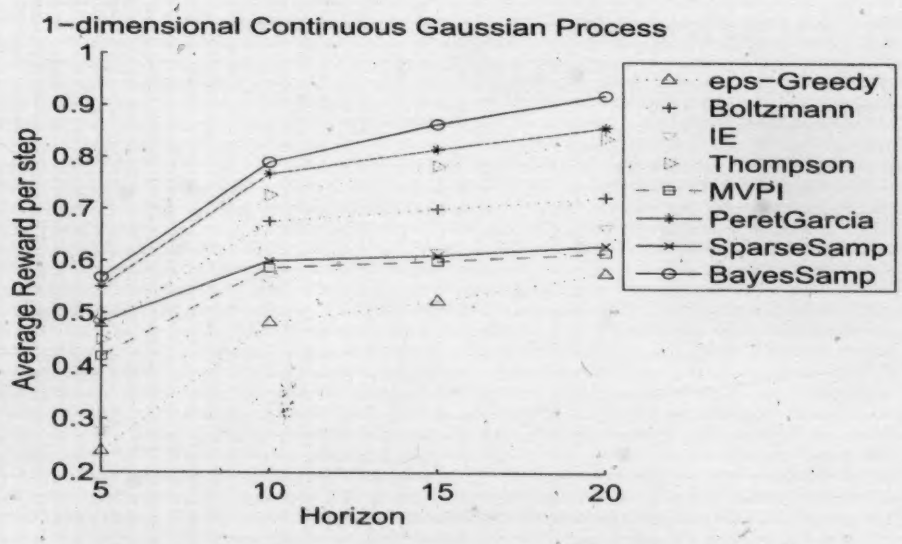

Figure 4.4: Results: 1-dimensional continuous action Gaussian process

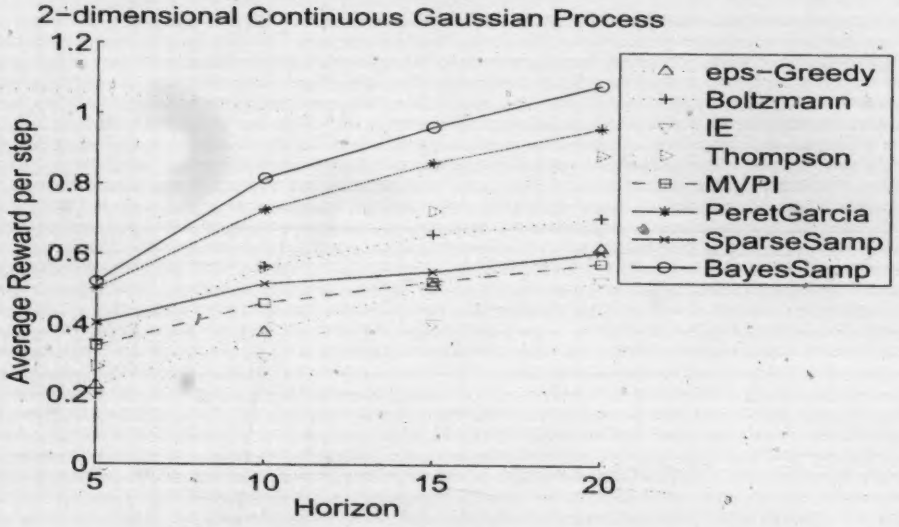

Figure 4.5: Results: 2-dimensional continuous action Gaussian process

More interesting domains involve actions where their rewards are correlated, and the number of actions is large or infinite. To examine those situations, I conducted experiments in a scenario that involved *continuous* action spaces. Specifically, I considered problems where the reward distribution over actions is defined by a *Gaussian process* prior over the action space (Williams, 1999). This creates an interesting exploration problem where rewards are correlated between actions, and actions themselves are not restricted to a trivial finite set.

Figures 4.4 and 4.5 show the results of the two continuous problems I considered. The first involved a 1-dimensional action space and the second a 2-dimensional action space. In each case, a Gaussian process prior over rewards was defined by an RBF kernel on actions, specifying the covariance between action rewards. (I used a Gaussian RBF kernel with width parameter 1. The noise standard deviation was set to $\sigma = 0.5$.) Technically, sparse sampling and MPI are unable to cope with continuous action spaces, so I sampled actions for them to consider according to a uniform distribution. Figures 4.4 and 4.5 show a clear advantage for Bayesian sparse sampling over sparse sampling and the myopic approaches—using the same number of lookahead nodes as sparse sampling and Péret & Garcia sampling, and similar computation to MVPI. Surprisingly, Péret & Garcia sampling performed nearly as well in this case, even though it exhibits weaker performance in the bandit problems.

4.7 Conclusion

I have proposed a simple approach to improving action selection quality in model based Bayesian reinforcement learning. My approach provides a flexible framework for integrating the prior knowledge of the problem with the interactive experience of a decision maker.

I have observed that this approach achieved good performance over other action selection strategies, in particular for 2-dimensional continuous Gaussian process domain, where both action space and reward space are continuous. The main advantage of my approach is that it yields improved exploration/exploitation decision

making whenever Bayesian posteriors can be conveniently calculated. The main drawback of my approach is shared by all model based Bayesian approaches to reinforcement learning: the need to repeatedly solve an MDP planning problem. Nevertheless, there are many interesting domains where this is not a significant barrier, and promising approaches have been developed for mitigating this expense (Dearden et al., 1999).

In the next chapter, I am going to present an approximate POMDP planning algorithm.

Chapter 5

Quadratic Approximations for POMDP Planning

Partially observable Markov decision processes (POMDPs) are an intuitive and general way to model sequential decision making problems under uncertainty. Unfortunately, even approximate planning in POMDPs is known to be hard, and developing heuristic planners that can deliver reasonable results in practice has proved to be a significant challenge.

In this chapter, I present a new approach to approximate value-iteration for POMDP planning that is based on quadratic rather than piecewise linear function approximators. Specifically, I approximate the optimal value function by a convex upper bound composed of a fixed number of quadratics, and optimize it at each stage by semidefinite programming. I demonstrate that this approach can achieve competitive approximation quality to current techniques while still maintaining a bounded size representation of the function approximator. Moreover, an upper bound on the optimal value function can be preserved if required. Overall, the technique requires computation time and space that is only linear in the number of decision stages.

This chapter is organized as follows. Section 5.1 motivates the problem. Section 5.2 introduces the POMDP model. Section 5.3 surveys the value function approaches. Section 5.5 demonstrates my approximate algorithm on benchmark POMDP problems.

5.1 Motivation

Partially observable Markov decision processes (POMDPs) are a general model of an agent acting in an environment, where the effects of the agent's actions and the observations it can make about the current state of the environment are both subject to uncertainty. The agent's goals are specified by rewards it receives (as a function of the states it visits and actions it executes), and an optimal behavior strategy in this context chooses actions, based on the history of observations, that maximizes the long term reward of the agent.

POMDPs have become an important modeling formalism in robotics and autonomous agent design (Thrun et al., 2005; Pineau et al., 2003c). Much of the current work on robot navigation and mapping, for example, is now based on stochastic transition and observation models (Thrun et al., 2005; Roy et al., 2005). Moreover, POMDP representations have also been used to design autonomous agents for real world applications, including nursing (Pineau et al., 2003c) and elderly assistance (Boger et al., 2005).

Despite their convenience as a modeling framework however, POMDPs pose difficult computational problems. It is well known that solving for optimal behavior strategies or even just approximating optimal strategies in a POMDP is intractable (Madani et al., 2003; Mundhenk et al., 2000). Therefore, a lot of work has focused on developing heuristics for computing reasonable behavior strategies for POMDPs. These approaches have generally followed three broad strategies: value function approximation (Hauskrecht, 2000; Spaan & Vlassis, 2005; Pineau et al., 2003b; Parr & Russell, 1995), policy based optimization (Ng & Jordan, 2000; Poupart & Boutilier, 2003; Poupart & Boutilier, 2004; Amato et al., 2006), and stochastic sampling (Kearns et al., 2002; Thrun, 2000). In this chapter, I focus on the value function approximation approach and contribute a new perspective to this strategy.

Most previous work on value function approximation for POMDPs has focused on representations that explicitly maintain a set of α-vectors or belief states. This is motivated by the fact that the optimal value function, considered as a function of the

belief state, is determined by the maximum of a set of linear functions—specified by α-vectors—where each α-vector is associated with a specific behavior policy. Since the optimal value function is given by the maximum of a (large) set of α-vectors, it is natural to consider approximating it by a subset of α-vectors, or at least a small set of linear functions. In fact, even an exact representation of the optimal value function need not keep every α-vector, but only those that are maximal for at least some "witness" belief state. Motivated by this characterization, most value function approximation strategies attempt to maintain a smaller subset of α-vectors by focusing on a reduced set of belief states (Spaan & Vlassis, 2005; Hauskrecht, 2000; Pineau et al., 2003b). Although much recent progress has been made on α-vector based approximations, a drawback of this approach is that the number of α-vectors stored generally has to grow with the number of value iterations to maintain an adequate approximation (Pineau et al., 2003b).

In this chapter, I consider an alternative approach that drops the notion of an α-vector entirely from the approximation strategy. Instead I exploit the other fundamental observation about the nature of the optimal value function: since it is determined by a belief-state-wise maximum over linear functions, the optimal value function must be a convex function of the belief state (Sondik, 1978; Boyd & Vandenberghe, 2004). My strategy, then, is to compute a convex approximation to the optimal value function that is based on quadratic rather than linear functions of the belief state. The advantage of using a quadratic basis for value function approximation is several-fold: First, the size of the representation does not have to grow merely to model an increasing number of facets in the optimal solution; thus one can keep a bounded size representation at each horizon. Second, a quadratic representation allows one to conveniently maintain a provable upper bound on the optimal values in an explicit compact representation without requiring auxiliary linear programming to be used to retrieve the bound, as in current grid based approaches (Hauskrecht, 2000; Smith & Simmons, 2005). Third, the computational cost of updating the approximation does not change with iteration number (either in time or space), so the overall computation time is only linear in the horizon. Finally, as I demonstrate below, despite a significant reduction in representation size, convex

quadratics are still able to achieve competitive approximation quality on benchmark POMDP problems.

5.2 Partially Observable Markov Decision Processes

I begin with Markov decision processes (MDP) since I will need to exploit some basic concepts from MDPs in my approach below. Recall that an MDP is defined by a set of states S, a set of actions A, a state transition model $p(s'|s,a)$, and a reward model $\mathcal{R}(s,a)$. In this setting, a deterministic policy is specified by a function from states to actions, $\pi : S \rightarrow A$, and the *value function* for a policy is defined as the expected future discounted reward the policy obtains from each state

$$V^{\pi}(s) = \mathrm{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \pi(s_t)) \,\middle|\, s_0 = s\right]$$

Here the discount factor, $0 \leq \gamma < 1$, expresses a tradeoff between short term and long term reward. It is known that there exists a deterministic optimal policy whose value function dominates all other policy values in every state (Bertsekas, 1995). This optimal value function also satisfies the Bellman equation

$$V^*(s) = \max_a \mathcal{R}(s,a) + \gamma \sum_{s'} p(s'|s,a) V^*(s') \tag{5.1}$$

Computing the optimal value function for a given MDP can be accomplished in several ways. The two ways I consider below are *value iteration* and *linear programming*. Value iteration is based on repeatedly applying the Bellman backup operator, $V_{n+1} = \mathcal{H}V_n$, specified by

$$V_{n+1}(s) = \max_a \mathcal{R}(s,a) + \gamma \sum_{s'} p(s'|s,a) V_n(s') \tag{5.2}$$

It can be shown that $V_n \rightarrow V^*$ in the L_∞ norm, and thus V^* is a fixed point of (5.2) (Bertsekas, 1995). V^* is also the solution to the linear program

$$\min_V \sum_s V(s) \text{ s.t. } V(s) \geq \mathcal{R}(s,a) + \gamma \sum_{s'} p(s'|s,a) V(s') \tag{5.3}$$

for all $s \in S$ and $a \in A$. It turns out that for *continuous* state spaces, the Bellman equation (see Equation 5.1) still characterizes the optimal value function, replacing

the transition probabilities with conditional densities and the sums with Lebesgue integrals. However, computationally, the situation is not so simple for continuous state spaces, since the integrals must now somehow be solved in place of the sums, and Equation 5.3 is no longer finitely defined. Nevertheless, continuous state spaces are unavoidable when one considers POMDP planning.

POMDPs extend MDPs by introducing an observation model $p(o'|a, s')$ that governs how a noisy observation $o' \in \mathcal{O}$ is related to the underlying state s' and the action a. Having access to only noisy observations of the state complicates the problem of choosing optimal actions significantly. The agent now never knows the exact state of the environment, but instead must infer a distribution over possible states, $b(s)$, from the history of observations and actions. Nevertheless, given an action a and observation o' the agent's *belief state* can be easily updated by Bayes' rule

$$b'_{(b,a,o')}(s') \;=\; p(o'|a, s') \sum_s p(s'|s, a) b(s)/Z \tag{5.4}$$

where $Z = p(o'|b, a) = \sum_{s'} p(o'|a, s') \sum_s p(s'|s, a) b(s)$.

By the Markov assumption, the belief state is a sufficient representation upon which an optimal behavior strategy can be defined (Sondik, 1978). Therefore, a policy is naturally specified in this setting by a function from belief states to actions, $\pi : B \rightarrow A$, where B is the set of all possible distributions over the underlying state space S (an $|S| - 1$ dimensional simplex). Obviously for any environment with two or more states there are an infinite number of belief states, and not every policy can be finitely represented. Nevertheless, one can still define the value function of a policy as the expected future discounted reward obtained from each belief state

$$V^\pi(b) \;=\; \mathrm{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(b_t, \pi(b_t)) \,\middle|\, b_0 = b \right]$$

where $r(b, a) = \sum_s \mathcal{R}(s, a) b(s)$. Thus, a POMDP can be treated as an MDP over belief states; that is, a continuous state MDP. As before, an optimal policy obtains the maximum value for each belief state, and its value function satisfies the Bellman

equation (Sondik, 1978)

$$V^*(b) = \max_a r(b,a) + \gamma \sum_{b'} p(b'|b,a) V^*(b')$$
$$= \max_a r(b,a) + \gamma \sum_{o'} p(o'|b,a) V^*(b'_{(b,a,o')}) \quad (5.5)$$

Unfortunately, solving the above functional equation for V^* is hard. Known techniques for computing the optimal value function are generally based on *value iteration* (Cassandra et al., 1997; Zhang & Zhang, 2001); although policy based approaches are also possible (Sondik, 1978; Poupart & Boutilier, 2003; Poupart & Boutilier, 2004). As above, value iteration is based on repeatedly applying a Bellman backup operator, $V_{n+1} = \mathcal{H}V_n$, to a current value function approximation. In this case, a current lower bound, V_n, is represented by a finite set of α-vectors, $\Gamma_n = \{\alpha_{\pi'} : \pi' \in \Pi_n\}$, where each α-vector is associated with an n-step behavior strategy π'. Given Γ_n, the value function is represented by

$$V_n(b) = \max_{\alpha_{\pi'} \in \Gamma_n} b \cdot \alpha_{\pi'}$$

At each stage of value iteration, the current lower bound is updated according to the Bellman backup, $V_{n+1} = \mathcal{H}V_n$, such that

$$V_{n+1}(b) = \max_a r(b,a) + \gamma \sum_{o'} p(o'|b,a) V_n(b'_{(b,a,o')}) \quad (5.6)$$
$$= \max_a b \cdot r_a + \gamma \sum_{o'} b \cdot \arg_{g_{(\pi',a,o')}} \max_{\pi' \in \Pi_n} b \cdot g_{(\pi',a,o')}$$
$$= \max_{\{o' \to \pi'\}} b \cdot \alpha_{a,\{o' \to \pi'\}} \quad (5.7)$$

where I use the quantities

$$g_{(\pi',a,o')}(s) = \sum_{s'} p(o'|a,s') p(s'|a,s) \alpha_{\pi'}(s')$$
$$\alpha_{a,\{o' \to \pi'\}} = r_a + \gamma \sum_{o'} g_{(\pi',a,o')}$$

Once again it is known that $V_n \to V^*$ in the L_∞ norm, and thus V^* is a fixed point of Equation 5.6 (Sondik, 1978).

Although the size of the representation for V_{n+1} remains finite, it can be exponentially larger than V_n in the worst case, since enumerating every possibility for

$a, \{o' \rightarrow \pi'\}$ over $a \in A$, $o \in O$, $\pi' \in \Pi_n$, yields $|\Pi_{n+1}| \leq |A||\Pi_n|^{|O|}$ combinations. Many of these α-vectors are not maximal for any belief state, and can be pruned by running a linear program for each that verifies whether there is a witness belief state for which it is maximal (Cassandra et al., 1997). Thus, the set of α-vectors, Γ_n, action strategies, Π_n, and witness belief states, B_n, are all associated 1 to 1. However, even with pruning, exact value iteration cannot be run for many steps, even on small problems.

5.3 Value Function Approximation Strategies

Much research has focused on approximating the optimal value function, aimed for the most part at reducing the time and space complexity of the value iteration update. Work in this area has considered various strategies (Hauskrecht, 2000), including direct MDP approximations and variants, and using function approximation to fit V_{n+1} over sampled belief states (Parr & Russell, 1995; Littman et al., 1995). However, two approaches have recently become the most dominant: grid based and belief point approximations.

The grid based approach (Gordon, 1995; Hauskrecht, 2000; Zhou & Hansen, 2001; Bonet, 2002) maintains a finite collection of belief states along with associated value estimates $\{\langle b, \bar{V}_n(b)\rangle : b \in B_{grid}\}$. These value estimates are updated by applying the Bellman update on $b \in B_{grid}$. An important advantage of this approach is that it can maintain an upper bound on the optimal value function. Unfortunately, maintaining a tight bound entails significant computational expense (Hauskrecht, 2000): First, B_{grid} must contain all corners of the simplex so that its convex closure spans B. Second, each successor belief state b' in Equation 5.6 must have its interpolated value estimate minimized by a linear program (Zhou & Hansen, 2001). Below I show that this large number of linear programs can be replaced with a single convex optimization.

Unlike the grid based approach, which takes a current belief state in B_{grid} and projects it forward to belief states outside of B_{grid}, the belief point approach only considers belief states in a witness set B_{wit} (Pineau et al., 2003b; Smith & Sim-

mons, 2005). Specifically, the belief point approximation maintains a lower bound by keeping a subset of α-vectors associated with these witness belief states. To further explain this approach, let $\Gamma_n = \{\alpha_{\pi'} : \pi' \in \hat{\Pi}_n\}$, so that there is a 1 to 1 correspondence between α-vectors in Γ_n, action strategies in $\hat{\Pi}_n$ and belief states in B_{wit}. Then the set of α-vectors is updated by applying the Bellman backup, but restricting the choices in Equation 5.7 to $\pi' \in \hat{\Pi}_n$, and only computing Equation 5.7 for $b \in B_{wit}$. Thus, the number of α-vectors in each iteration remains bounded and associated with $b \in B_{wit}$.

The quality of both these approaches is strongly determined by the sets of belief points, B_{grid} and B_{wit}, they maintain. For the belief point approach, one generally has to grow the number of belief points at each iteration to maintain an adequate bound on the optimal value function. (Pineau et al., 2003b) suggested doubling the size at each iteration, but recently a more refined approach was suggested by (Smith & Simmons, 2005).

5.4 Convex Quadratic Upper Bounds

The key observation behind my proposed approach is that one does not need to be confined to piecewise linear approximations. The intuition is that convex quadratic approximations are particularly well suited for value function approximation in POMDPs. This is motivated by the fact that each value iteration step produces a maximum over a set of convex functions, yielding a result that is always convex. Thus, one can plausibly use a convex quadratic function to upper bound the maximum over α-vectors, and more generally to upper bound the maximum over any set of back-projected convex value approximations from iteration n. The basic goal then is to retain a compact representation of the value approximation by exploiting the fact that quadratics can be more efficient at approximating a convex upper bound than a set of linear functions (see Figure 5.1). As with piecewise linear approximations, the quality of the approximation can be improved by taking a maximum over a set of convex quadratics, which would yield a convex piecewise quadratic rather than piecewise linear approximation. In this chapter, however, I will focus on the

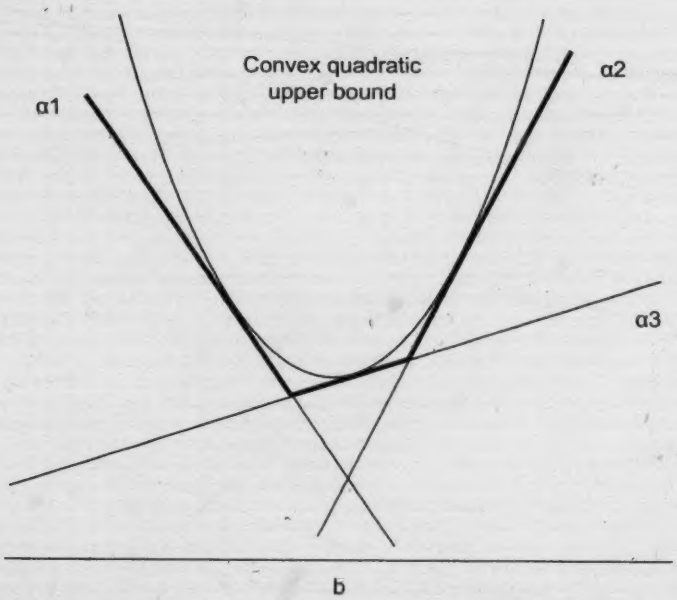

Figure 5.1: Illustration of a convex quadratic upper bound approximation to a maximum of linear functions $b \cdot \alpha_\pi$.

most naive choice, and approximate the value function with a *single* quadratic in each step of value iteration. The subsequent extension to multiple quadratics is discussed below.

An important advantage the quadratic form has over other function approximation representations is that it permits a convex minimization of the upper bound, as I demonstrate below. Such a convenient formulation is not readily achievable for other function representations. Also, since one is not compelled to grow the size of the representation at each iteration, one can obtain an approach that runs in linear time in the number of value iteration steps.

There are a few drawbacks in dropping the piecewise linear representation however. One drawback is that I lose the 1 to 1 correspondence between α-vectors and behavior strategies π', which means that greedy action selection requires a one step look ahead calculation based on Equation 5.5. The second drawback is that the convex optimization problem I have to solve at each value iteration is more complex than a simple linear program.

5.4.1 Convex Upper Bound Iteration

The main technical challenge I face is to solve for a tight quadratic upper bound on the value function at each stage of value iteration. Interestingly, this can be done effectively with a convex optimization as follows. I represent the value function approximation over belief states by a quadratic form

$$\hat{V}_n(b) = b^\top W_n b + w_n^\top b + \omega_n \tag{5.8}$$

where W_n is a square matrix of weights, w_n is a vector of weights, and ω_n is a scalar offset weight. Equation 5.8 defines a *convex* function of belief state b if and only if the matrix W_n is positive semidefinite (Boyd & Vandenberghe, 2004). I denote the semidefinite constraint on W_n by $W_n \succeq 0$. As shown above, one step of value iteration involves expanding (and back-projecting) a value approximation from stage n; defining the value function at stage $n+1$ by the maximum over the expanded, back-projected set. However, back-projection entails some additional complication in this case because I do not maintain a set of α-vectors, but rather maintain a quadratic function approximation at stage n. That is, the approximate value iteration step has to pull the quadratic form through the backup operator. Unfortunately, the result of a backup is no longer a quadratic, but a rational (quadratic over linear) function. Fortunately, however, the result of this backup is still convex, as I now show.

Let the action-value backup of $\hat{V}$ be denoted by

$$q_a(b) = r(b,a) + \gamma \sum_{o'} p(o'|b,a) \hat{V}(b'_{(b,a,o')}) \tag{5.9}$$

To express this as a function of b, I need to expand the definitions of $b'_{(b,a,o')}$ and $\hat{V}_n$ respectively. First, note that $b'_{(b,a,o')}$ is a ratio of a vector linear function of b over a scalar linear function of b by Equation 5.4, therefore one can represent it by

$$b'_{(b,a,o')} = \frac{M_{a,o'} b}{p(o'|b,a)} = \frac{M_{a,o'} b}{\mathbf{1}^\top M_{a,o'} b} \tag{5.10}$$

where $M_{a,o'}$ is a matrix such that $M_{a,o'}(s',s) = p(o'|a,s')p(s'|s,a)$, and $\mathbf{1}$ denotes

a vector of all 1s. Substituting Equations 5.8 and 5.10 into Equation 5.9 yields

$$q_a(b) = \underbrace{r(b,a)}_{\text{scalar}} + \underbrace{\gamma \sum_{o'} \frac{b^\top M_{a,o'}^\top W M_{a,o'} b}{\mathbf{1}^\top M_{a,o'} b}}_{\text{convex}} + \underbrace{(w+\omega\mathbf{1})^\top M_{a,o'} b}_{\text{linear}} \quad (5.11)$$

Theorem 8 *$q_a(b)$ is convex in b.*

Proof: Since $W \succeq 0$, we have $M_{a,o'}^\top W M_{a,o'} \succeq 0$. Therefore, $M_{a,o'}^\top W M_{a,o'} = UU^\top$ for some U. Letting $f(b) \triangleq \frac{b^\top M_{a,o'}^\top W M_{a,o'} b}{\mathbf{1}^\top M_{a,o'} b}$, this quadratic-over-linear function can be written as

$$f(b) = \frac{b^\top UU^\top b}{v^\top b} = (U^\top b)^\top (v^\top bI)^{-1} (U^\top b) \quad (5.12)$$

where $v^\top b = \mathbf{1}^\top M_{a,o'} b$. Since $M_{a,o'} > 0$ and $b > 0$, $v^\top bI \succ 0$ holds.

Recall that a function is convex iff its epigraph is a convex set (see Section 3.1.7 in (Boyd & Vandenberghe, 2004)). By the definition of the epigraph of a function (Boyd & Vandenberghe, 2004),

$$\mathbf{epi} f = \{(b, v^\top bI, \delta) \mid v^\top bI \succ 0, (U^\top b)^\top (v^\top bI)^{-1} (U^\top b) \leq \delta\} \quad (5.13)$$

By the *Schur complement* lemma, [1] we have $\begin{bmatrix} v^\top bI & U^\top b \\ (U^\top b)^\top & \delta \end{bmatrix} \succeq 0$. Then the epigraph of $f(b)$ becomes

$$\mathbf{epi} f = \left\{ (b, v^\top bI, \delta) \mid v^\top bI \succ 0, \begin{bmatrix} v^\top bI & U^\top b \\ (U^\top b)^\top & \delta \end{bmatrix} \succeq 0 \right\} \quad (5.14)$$

$\mathbf{epi} f$ is convex because this set can be written as a linear matrix inequality.[2] Therefore, we have $f(b)$ is convex.

Thus, $q_a(b)$ is convex in b since $q_a(b)$ is a sum of a scalar, a convex function in b, and a linear function in b. ∎

Corollary 1 *Given a convex quadratic representation for $\hat{V}_n$, $\max_a q_a(b)$, and hence $\mathcal{H}\hat{V}_n$, is convex in b.*

[1] If $v^\top bI \succ 0$, then $\begin{bmatrix} v^\top bI & U^\top b \\ (U^\top b)^\top & \delta \end{bmatrix} \succeq 0$ iff $\delta - (U^\top b)^\top (v^\top bI)^{-1} (U^\top b) \geq 0$ (see Appendix A.5.5 in (Boyd & Vandenberghe, 2004)).

[2] See Example 2.10 in (Boyd & Vandenberghe, 2004).

So back-projecting the convex quadratic representation still yields a convex result. The goal is to optimize a tight quadratic upper bound on the maximum of these convex functions (which of course is still convex). In some approaches below I will use the back-projected action-value functions directly. However, in other cases, it will prove advantageous if one can work with linear upper bounds on the back-projections.

Proposition 1 *The tightest linear upper bound on $q_a(b)$ is given by $q_a(b) \leq u_a^\top b$ for a vector u_a such that $u_a^\top 1_s = q_a(1_s)$ for each corner belief state 1_s.*

5.4.2 Algorithmic Approach

I would like to solve for a quadratic $\hat{V}_{n+1}$ at stage $n+1$ that obtains as tight an upper bound on $\mathcal{H}\hat{V}_n$ as possible. To do this, one appeal to the linear program characterization of the optimal value function (see Equation 5.3) which also is expressed as minimizing an upper bound on the back-projected value function. Unfortunately, here, since one is no longer working with a finite space, one cannot formulate a linear program but rather have to pose a generalized semi-infinite program

$$\min_{W,w,\omega} \int_b \left(b^\top W b + w^\top b + \omega\right) \mu(b)\, db \quad \text{subject to} \qquad (5.15)$$
$$b^\top W b + w^\top b + \omega \geq q_a(b), \quad \forall\, a, b; \quad W \succeq 0$$

where $\mu(b)$ is a measure over the space of possible belief states. This semi-infinite program specifies a linear objective subject to linear constraints (albeit infinitely many linear constraints); and hence is a *convex* optimization problem in W, w, ω.

There are two main difficulties in solving this convex optimization problem. First, the objective involves an integral with respect to a measure $\mu(b)$ on belief states. This measure is arbitrary (except that it must have full support on the belief space B) and allows one to control the emphasis the minimization places on different regions of the belief space. For simplicity, I assume the measure is a Dirichlet distribution, specified by a vector of prior parameters $\theta(s), \forall s \in S$. The Dirichlet distribution is particularly convenient in this context since one can specify a uniform distribution over the belief simplex merely by setting $\theta(s) = 1$ for all s. Moreover,

the required integrals for the Dirichlet have a closed form solution, which allows us to simply *precompute* the linear coefficients for the weight parameters, by

$$\int_b \left(b^\top W b + w^\top b + \omega\right) \mu(b) db \;=\; \langle W, \mathrm{E}[bb^\top]\rangle + w^\top \mathrm{E}[b] + \omega$$

where $\mathrm{E}[b] = \theta/\|\theta\|_1$; $\mathrm{E}[bb^\top] = (\mathrm{diag}(\mathrm{E}[b]) + \|\theta\|_1 \mathrm{E}[b]\mathrm{E}[b]^\top)/(1 + \|\theta\|_1)$ (Gelman et al., 1995); and $\langle A, B\rangle = \sum_{ij} A_{ij} B_{ij}$. That is, one can specify θ and compute the linear coefficients ahead of time.

The second and more difficult problem with solving Equation 5.15 is to find a way to cope with the infinite number of linear constraints. Here, I address the problem with a straightforward constraint generation approach. The idea is to solve Equation 5.15, iteratively, by keeping a finite set of constraints, each corresponding to a belief state, and solving the finite *semidefinite* program

$$\begin{aligned} &\min_{W,w,\omega} \langle W, E[bb^\top]\rangle + w^\top E[b] + \omega \quad \text{subject to} \\ &b_i^\top W b_i + w^\top b_i + \omega \geq q_a(b_i), \;\; \forall\, a, b_i \in \mathcal{C}; \;\; W \succeq 0 \end{aligned} \tag{5.16}$$

Given a putative solution, W, w, ω, a new constraint can be obtained by finding a belief state b that solves

$$\begin{aligned} &\min_b b^\top W b + w^\top b + \omega - q_a(b) \quad \text{subject to} \\ &\qquad b \geq 0, \;\; \textstyle\sum_s b(s) = 1 \end{aligned} \tag{5.17}$$

for each a. If the minimum value is nonnegative for all a then there are no violated constraints and I have a solution to Equation 5.15.

Unfortunately, Equation 5.17 cannot directly be used for constraint generation, since $q_a(b)$ is a convex function of b (see Theorem 8) and hence $-q_a(b)$ is concave; yielding a non-convex objective. Thus, to use Equation 5.17 for constraint generation I need to follow an alternative approach. I propose three different approaches to this problem.

The first strategy maintains a provable upper bound on the optimal value function by strengthening the constraint threshold with the linear upper bound $u_a^\top b \geq q_a(b)$ from Proposition 1. Replacing $q_a(b)$ with $u_a^\top b$ (in Equations 5.15 and 5.17) ensures that an upper bound will be maintained, but also reduces Equation 5.17 to

a quadratic program that can be efficiently minimized to produce a belief state with maximum constraint violation.

The second strategy relaxes the upper bound guarantee by only substituting $u_a^\top b$ for $q_a(b)$ in the constraint generation procedure, maintaining an efficient quadratic programming formulation there, but keeping $q_a(b)$ in the main optimization (see Equation 5.16). This no longer guarantees an upper bound, but can still produce better approximations in practice because the bounds do not have to be artificially strengthened.

The third strategy side-steps optimal constraint generation entirely, and instead chooses a fixed set of belief states for the constraint set C in Equation 5.16. In this way, the semidefinite program (see Equation 5.16) needs to be solved only once per value iteration step. This strategy doesn't produce an upper bound either but the resulting approximation is fast and effective in practice.

Finally, to improve approximation quality, one could augment the approximate value function representation with a maximum over a set of quadratics, much as with α-vectors. One natural way to do this would be to maintain a separate quadratic for each action, a, in Equation 5.15.

5.5 Experimental Results

I implemented the proposed approach using SDPT3 (Toh et al., 1999) as the semidefinite program solver for solving the finite semidefinite program in Equation 5.16. Specifically, in my experiments, I have investigated the third (simplest) strategy mentioned above, which only used a random sample of belief states to specify the constraints in the constraint set C. I compared this method to two current value function approximation strategies in the literature: Perseus (Spaan & Vlassis, 2005) and PBVI (Pineau et al., 2003b). Here, both Perseus and PBVI were run with the number of belief states fixed at 1000, whereas the convex quadratic method, CQUB, was run with 100 random belief states.

In my experiments, I considered the benchmark problems: [3] Maze (Hauskrecht,

[3] http://www.cassandra.org/pomdp/examples

1997), Tiger-grid, Hallway, Hallway2, and Aircraft. Table 5.1 gives the problem characteristics. In each case, a number of value iteration steps was fixed as shown in Table 5.1, and each method was run 10 times to generate an estimate of value function approximation quality.

Table 5.2 shows the results obtained by the various value function approximation strategies on these domains, reporting the expected discounted reward obtained by the greedy policies defined with respect to the value function estimates, as well as the average time and the size of the value function approximation.[4] Interestingly, the convex quadratic strategy CQUB performed surprisingly well in these experiments, competing with state of the art value function approximations while only using 100 random belief states for constraint generation in Equation 5.16. The policies that computed from CQUB are competitive with other state of the art techniques (Perseus and PBVI). The result is significantly stronger in the Hallway domains, reasonable good in the Maze and Aircraft domain, and slightly weaker in the Tiger-grid domain. The size of the representation for value function (i.e., the number of unknown variables) requires by CQUB is significantly less due to the representational power of quadratics. That is, convex quadratics capture value function structure more efficiently than the linear approximation approaches. Moreover, the size of representation in CQUB grows much slower than the ones in Persus and PBVI. However, the running time of CQUB is significant higher than previous techniques due to the need to solve a semidefinite program for every back projection of the value function.

5.6 Conclusion

I have introduced a new approach to value function approximation for POMDPs that is based on a convex quadratic bound rather than a piecewise linear approximation. The unknown parameters in the quadratics are optimized at each decision making stage by semidefinite programming.

From the empirical studies on the benchmark problems, I have found that quadratic

[4]For Perseus and PBVI, the size is $|S|$ times the number of α-vectors. For CQUB, the size is just $|S|(|S|+1)/2+|S|+1$, which corresponds to the number of variables in the quadratic approximator.

Problems	$\|S\|$	$\|A\|$	$\|O\|$	value iters
Maze	20	6	8	40
Tiger-grid	33	5	17	76
Hallway	57	5	21	55
Hallway2	89	5	17	33
Aircraft	100	10	31	10

Table 5.1: Problem characteristics.

	CQUB	Perseus	PBVI
Maze			
Avg. reward	45.35 $\pm$3.28	30.49 $\pm$0.75	46.70 $\pm$2.0
Run time (s)	197.71	60.00	0.66
Size	231	460	1160
Tiger-grid			
Avg. reward	2.16 $\pm$0.02	2.34 $\pm$0.02	2.25 $\pm$0.06
Run time (s)	7.5×10^3	61.36	28.47
Size	595	4422	15510
Hallway			
Avg. reward	0.58 $\pm$0.14	0.51 $\pm$0.06	0.53 $\pm$0.03
Run time (s)	7.5×10^3	61.26	39.79
Size	1711	3135	4902
Hallway2			
Avg. reward	0.43 $\pm$0.25	0.34 $\pm$0.16	0.35 $\pm$0.03
Run time (s)	1.8×10^4	63.72	27.97
Size	4095	4984	8455
Aircraft			
Avg. reward	16.70 $\pm$0.58	12.73 $\pm$4.63	16.37 $\pm$0.42
Run time (s)	3.8×10^5	60.01	8.03
Size	5151	10665	47000

Table 5.2: Experimental results. Mean discounted reward obtained over 1000 trajectories using the greedy policy for each value function approximation, averaged over 10 runs of value iteration.

approximators can achieve highly competitive approximation quality without growing the size of the representation, even while explicitly focusing on a tiny fraction of the belief states. In summary, CQUB maintains a compact representation of the value function and only requires computational time and space linear in the number of decision stages. The drawback of CQUB is that it took *longer* run time than Perseus or PBVI in my runs above, which mainly comes from the limitation of the speed of the current semi-definite program solver.

Since this approach introduces a new perspective on approximating POMDP planning, I hope that it can lead to new avenues of research in value approximation for POMDPs.

Chapter 6

Conclusions

This thesis addresses the challenge of scaling up algorithms for sequential decision making under uncertainty. Specifically, my research attempts to tackle three key outstanding issues in sequential decision making in uncertain environments: combining off-policy updates with generalization, balancing exploration and exploitation, and handling partial observability of the environment.

6.1 Contributions

The key contributions of this thesis can be summarized as follows.

Dual representations and algorithms (Chapter 3). I have introduced dual representations and a body of novel dual algorithms for sequential decision making problems. Dual representations maintain an explicit representation of stationary distributions as opposed to value functions. Therefore, dual algorithms, since they are based on estimating normalized probability distributions rather than unbounded value functions, avoid divergence even in the presence of approximation and off-policy updates. Moreover, dual algorithms remain stable in situations where standard value function estimation diverges. Furthermore, this dual view offers a coherent and comprehensive perspective on optimal sequential decision making problems, provides an alternative to standard value function based techniques, and opens new avenues for solving sequential decision making problems. I studied the convergence properties of the dual dynamic programming algorithms both theoretically and empirically. I observed that on-policy updates converge in both primal and dual

dynamic programming algorithms while off-policy updates diverge when composed with gradient operator in the primal, not in the dual. Learning from the techniques for proving convergence for the primal algorithms, I proved the convergence of tabular on-policy ($\mathcal{O}$), tabular off-policy ($\mathcal{M}$), and projection on-policy ($\mathcal{PO}$) updates for the dual. The reason is that these updates are non-expansion (or contraction) operators with respect to the right norm.

A new technique for on-line action selection (Chapter 4). I have proposed a simple approach to improving action selection quality in model based Bayesian reinforcement learning. The technique exploits information in a Bayesian posterior to make intelligent actions by growing an adaptive, sparse lookahead tree. I demonstrate that Bayesian sparse sampling improves action selection in simple reinforcement learning scenarios. The main advantage is that the approach yields improved exploration/exploitation decision making whenever Bayesian posteriors can be conveniently calculated. The main drawback of my approach is shared by all model based Bayesian approaches to reinforcement learning: the need to repeatedly solve an MDP planning problem.

A novel algorithm for approximate POMDP planning (Chapter 5). Finally, I have introduced a new approach to value function approximation for POMDPs that is based on a convex quadratic bound rather than a piecewise linear approximation. Specifically, I approximate the optimal value function by a convex upper bound composed of a fixed number of quadratics, and optimize it by semidefinite programming. I have found that quadratic approximators can achieve highly competitive approximation quality without growing the size of the representation, even while explicitly focusing on only a tiny fraction of the belief states. Overall, the technique requires computation time and space that is only linear in the number of decision stages. However, CQUB took *longer* run time than Perseus or PBVI in my runs above. This drawback of my approach mainly comes from the limitation of the speed of the current semi-definite program solver. Still since this approach introduces a new perspective on approximating POMDP planning, I expect that this

approach can lead to new avenues of research in value approximation for POMDPs.

6.2 Future Research

There are a number of future research directions suggested by the work in this thesis.

Exploiting the joint primal-dual view of DP/RL. The dual view offers a coherent and comprehensive perspective on optimal sequential decision making problems, provides a viable alternative to standard value function based techniques, and opens new avenues for solving sequential decision making problems. Nevertheless, a problem can be solved jointly by a combination of both the primal and dual approaches.

Extensions of Bayesian sparse sampling. One possible future research for the Bayesian sparse sampling idea is to compare on-line action selection strategies with pre-compilation approaches (Boyan & Moore, 1996). It is also interesting to contemplate the prospect of hybrid action selection strategies that combine pre-compilation with on-line computation, perhaps by allowing a pre-compiled value function approximation to guide lookahead simulation without the need for on-line MDP planning.

Extensions of approximate POMDP planning. The quadratic approximation for POMDP planning can be extended by considering structured POMDP models, and combining value function approximation with policy based and sampling based approaches. Also the idea of belief state compression (Poupart & Boutilier, 2002; Poupart & Boutilier, 2004; Roy et al., 2005), and factored models (Boutilier & Poole, 1996; Feng & Hansen, 2001; Poupart, 2005) can help tackle POMDPs with large state spaces. Another idea I am exploring is the interpretation of convex quadratics as second order Taylor approximations to the optimal value function, which offers further algorithmic approaches with the potential for tight theoretical guarantees on approximation quality.

Combining Bayesian sparse sampling with approximation. The unification of Bayesian sparse sampling and quadratic approximate POMDP planning should provide a principled methodology for integrating prior knowledge of the problem with the interactive experience of a decision maker. Although the action selection strategy (Bayesian sparse sampling) in Chapter 4 is proposed for MDPs, the idea of making use of both prior knowledge and interactive experience by growing a sparse lookahead tree can be applied to sequential decision making in general. Potentially, the Bayesian sparse sampling idea can be extended to handle action selection in partially observable environments. However, the extension is non-trivial because a belief state in a POMDP is continuous. Fortunately, approximate POMDP planning algorithm in Chapter 5 can be a natural choice for computing the values of the leaf nodes in the tree evaluation. Combing Bayesian sparse sampling with convex upper bound iteration algorithm can give us a general framework to reflect a decision maker's the domain knowledge while allow it to behave adaptively as it can learn from its interactive experience.

Robotics applications. A robotic application I am currently pursuing is designing a mobile Vendorbot (a wheeled Pioneer III robot). The goal of Vendorbot project is to provide a real world domain to evaluate and extend my ideas. Imagine a robot wandering around an office building (like ours) selling candy bars and coffee to the people who might want to have them without stepping out of their offices. In addition to taking Internet food requests via wireless communication, the robot can try to learn the preferences and habits of its customers from its observations and experiences. One candidate strategy for a robot to select action on-line (e.g., where to visit) will be the ideas mentioned in Chapter 4 of this thesis. The robot might choose to give high priority to the Internet orders that are either expected to have high payoff or be trustworthy. By saying that orders are "trustworthy", I mean that the orders are from customers who have never cheated the robot in the past by refusing to pay for the delivery. Although I have to face challenging problems besides decision making, e.g., robot navigation, I hope that Vendorbot project provides an opportunity to investigate the flexibility of Bayesian sparse sampling strategy.

Bibliography

Abbeel, P., Coates, A., Quigley, M., & Ng, A. (2007). An application of reinforcement learning to aerobatic helicopter flight. In B. Schölkopf, J. Platt and T. Hoffman (Eds.), *Advances in neural information processing systems 19*. Cambridge, MA: MIT Press.

Amato, C., Bernstein, D., & Zilberstein, S. (2006). Solving POMDPs using quadratically constrained linear programs. *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (1995). Gambling in a rigged casino: the adversarial multi-armed bandit problem. *Proceedings of the 36th Annual Symposium on Foundations of Computer Science* (pp. 322–331).

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. *International Conference on Machine Learning* (pp. 30–37).

Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research, 15*, 319–350.

Bellman, R. (1961). *Adaptive control processes*. Princeton.

Bernardo, J. M., & Smith, A. F. M. (2000). *Bayesian theory*. Wiley.

Bernstein, D. S. (2005). *Matrix mathematics: Theory, facts, and formulas with application to linear systems theory*. Princeton University Press.

Berry, D., & Fristedt, B. (1985). *Bandit problems: Sequential allocation of experiments*. Chapman Hall.

Bertsekas, D. (1995). *Dynamic programming and optimal control*, vol. 2. Athena Scientific.

Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming*. Athena Scientific.

Bertsimas, D., & Nino-Mora., J. (1996). Conservation laws, extended polymatroids and multiarmed bandit problems; a polyhedral approach to indexable systems. *Mathematics of Operations Research, 21*, 257–305.

Boger, J., Poupart, P., Hoey, J., Boutilier, C., Fernie, G., & Mihailidis, A. (2005). A decision-theoretic approach to task assistance for persons with dementia. *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*.

Bonet, B. (2002). An ϵ-optimal grid-based algorithm for partially observable Markov decision processes. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML).*

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research, 11*, 1–94.

Boutilier, C., & Poole, D. (1996). Computing optimal policies for partially observable decision processes using compact representations. *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI).*

Boyan, J., & Moore, A. (1996). Learning evaluation functions for large acyclic domains. *Proceedings ICML.*

Boyan, J. A. (1999). Least-squares temporal difference learning. *Proceedings 16th International Conference on Machine Learning* (pp. 49–56). Morgan Kaufmann, San Francisco, CA.

Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. *Advances in Neural Information Processing Systems 7* (pp. 369–376). Cambridge, MA: The MIT Press.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization.* Cambridge University Press.

Bradtke, S. J. (1993). Reinforcement learning applied to linear quadratic regulation. *Advances in Neural Information Processing Systems 5).* San Mateo, CA: Morgan Kaufmann.

Bradtke, S. J., & Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning, 22*, 33–57.

Brafman, R. I., & Tennenholtz, M. (2001). R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *Proceedings IJCAI* (pp. 953–958).

Cassandra, A., Littman, M., & Zhang, N. (1997). Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI).*

Dayan, P. (1993). Improving generalisation for temporal difference learning: The successor representation. *Neural Computation, 5*, 613–624.

de Farias, D. P., & Van Roy, B. (2000). On the existence of fixed points for approximate value iteration and temporal-difference learning. *Journal Optimization Theory and Applications, 105*, 589–608.

Dearden, R., Friedman, N., & Andre, D. (1999). Model based Bayesian exploration. *Proceedings UAI.*

Dearden, R., Friedman, N., & Russell, S. (1998). Bayesian Q-learning. *Proceedings AAAI.*

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification.* Wiley.

Duff, M. (2002). *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes*. Doctoral dissertation, U. Mass Amherst.

Engel, Y., Mannor, S., & Meir, R. (2003). Bayes meets Bellman: The Gaussian process approach to temporal difference learning. *Proceedings ICML*.

Feng, Z., & Hansen, E. A. (2001). Approximate planning for factored POMDPs. *Proceedings of the Sixth European Conference on Planning*.

Frostig, E., & Weiss, G. (1999). Four proofs of Gittins' multiarmed bandit theorem. *Applied Probability Trust*.

Gelman, A., Carlin, J., Stern, H., & Rubin, D. (1995). *Bayesian data analysis*. Chapman & Hall.

Geramifard, A., Bowling, M., & Sutton, R. S. (2006). Incremental least-squares temporal difference learning (pp. 356–361.).

Gittins, J. (1989). *Multi-armed bandit allocation indices*. Wiley.

Gordon, G. (1995). Stable function approximation in dynamic programming. *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*.

Gordon, G. J. (2000). Reinforcement learning with function approximation converges to a region. *NIPS* (pp. 1040–1046).

Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research (JAIR), 19*, 399–468.

Hauskrecht, M. (1997). Incremental methods for computing bounds in partially observable markov decision processes. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI)*.

Hauskrecht, M. (2000). Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research, 13*, 33–94.

Jordan, M. (Ed.). (1999). *Learning in graphical models*. MIT Press.

Kaelbling, L. P. (1994). Associative reinforcement learning: Functions in k-DNF. *Machine Learning, 15*, 279–298.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence, 101*, 99–134.

Kakade, B., Ng, S., & Schneider, A. (2003). Policy search by dynamic programming. *In Proceedings of Neural Information Processing Systems, NIPS 16*.

Kakade, S., & Langford, J. (2002). Approximately optimal approximate reinforcement learning. *Proceedings of International Conference on Machine Learning*.

Karoui, N. E., & Karatzas, I. (1993). General Gittins index processes in discrete time. *Proc Natl Acad Sci USA* (pp. 1232–1236).

Katehakis, M. N., & Veinott, A. F. (1987). The multi-armed bandit problem: Decomposition and computation. *Mathematics of Operations Research, 12*, 262–268.

Kearns, M., Mansour, Y., & Ng, A. (2001). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *JMLR*, 1324–1331.

Kearns, M., Mansour, Y., & Ng, A. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning, 49*, 193–208.

Kearns, M., & Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. *Proceedings ICML.*

Kleinberg, R. (2005). Nearly tight bounds for the continuum-armed bandit problem. *Advances in Neural Information Processing Systems 17.*

Kocsis, L., & Szepesvári, C. (2006). Bandit based monte-carlo planning. *Proceedings of the 17th European Conference on Machine Learning* (pp. 282–293).

Koller, D., & Parr, R. (1999). Computing factored value functions for policies in structured MDPs. *Proceedings IJCAI.*

Koller, D., & Parr, R. (2000). Policy iteration for factored MDPs. *Proceedings UAI.*

Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research, 4*, 1107–1149.

Littman, M., Cassandra, A., & Kaelbling, L. (1995). Learning policies for partially observable environments: scaling up. *Proceedings of the Twelfth International Conference on Machine Learning (ICML).*

Luce, D. (1959). *Individual choice behavior.* Wiley.

Lusena, C., Goldsmith, J., & Mundhenk, M. (2001). Nonapproximability results for partially observable Markov decision processes. *JAIR, 14*, 83–103.

Madani, O., Hanks, S., & Condon, A. (2003). On the undecidability of proabilistic planning and related stochastic optimization problems. *Artificial Intelligence, 147*, 5–34.

Marbach, P., & Tsitsiklis, J. (2000). Gradient-based optimization of Markov reward processes: Practical variants. Machine Learning, 2000.

Martin, J. (1967). *Bayesian decision problems and Markov chains.* Wiley.

Michie, D., & Chambers, R. A. (1968). Boxes: An experiment in adaptive control. In E. Dale and D. Michie (Eds.), *Machine intelligence 2*, 137–152. Oliver and Boyd.

Mickova, J. (2000). Stochastic scheduling with multi-armed bandits. Master's thesis, University of Melbourne, Australia.

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning, 13*, 103–130.

Mundhenk, M., Goldsmith, J., Lusena, C., & Allender, E. (2000). Complexity of finite-horizon Markov decision processes. *JACM, 47*, 681–720.

Neal, R. (Ed.). (1996). *Bayesian learning for neural networks.* Springer.

Ng, A., & Jordan, M. (2000). Pegasus: A policy search method for large MDPs and POMDPs. *Proceedings UAI.*

Ng, A., Parr, R., & Koller, D. (1999a). Policy search via density estimation. *Proceedings NIPS.*

Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., & Liang, E. (2004). Autonomous inverted helicopter flight via reinforcement learning. *International Symposium on Experimental Robotics.*

Ng, A. Y., Harada, D., & Russell, S. (1999b). Policy invariance under reward transformations: theory and application to reward shaping. *Proc. 16th International Conf. on Machine Learning* (pp. 278–287). Morgan Kaufmann, San Francisco, CA.

Parr, R., & Russell, S. (1995). Approximating optimal policies for partially observable stochastic domains. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI).*

Patrascu, R., Poupart, P., Schuurmans, D., Boutilier, C., & Guestrin, C. (2002). Greedy linear value-approximation for factored Markov decision processes. *AAAI.*

Péret, L., & Garcia, F. (2004). On-line search for solving Markov decision processes via heuristic sampling. *Proceedings ECAI* (pp. 530–534).

Pineau, J., Gordon, G., & Thrun, S. (2003a). Applying metric-trees to belief-point POMDPs. *Proceedings NIPS.*

Pineau, J., Gordon, G., & Thrun, S. (2003b). Point-based value iteration: An anytime algorithm for POMDPs. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI).*

Pineau, J., Montemerlo, M., Pollack, M., Roy, N., & Thrun, S. (2003c). Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems, 42,* 271–281.

Poupart, P. (2005). *Exploiting structure to efficienty solve large scale partially observable Markov decision processes.* Doctoral dissertation, Department of Computer Science, University of Toronto.

Poupart, P., & Boutilier, C. (2002). Value-directed compression of POMDPs. *Advances in Neural Information Processing Systems (NIPS 15).*

Poupart, P., & Boutilier, C. (2003). Bounded finite state controllers. *Advances in Neural Information Processing Systems (NIPS 16).*

Poupart, P., & Boutilier, C. (2004). VDCBPI: An approximate scalable algorithm for large POMDPs. *Advances in Neural Information Processing Systems (NIPS 17).*

Precup, D., Sutton, R. S., & Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. *Proc. 18th International Conf. on Machine Learning* (pp. 417–424). Morgan Kaufmann, San Francisco, CA.

Puterman, M. (1994). *Markov decision processes: Discrete dynamic programming.* Wiley.

Rasmussen, C. E., & Kuss, M. (2004). Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems 16* (pp. 751–759).

Ross, S. (1997). *Introduction to probability models*. Academic Press. 6th edition.

Roy, N., Gordon, G., & Thrun, S. (2005). Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research, 23*, 1–40.

Salganicoff, M., & Ungar, L. (1995). Active exploration and learning in real-valued spaces using multi-armed bandit allocation indices. *Proceedings ICML* (pp. 480–487).

Schuurmans, D., & Patrascu, R. (2001). Direct value-approximation for factored mdps. *Proceedings NIPS*.

Singh, S. P., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning, 38*, 287–308.

Smith, T., & Simmons, R. (2005). Point-based POMDP algorithms: Improved analysis and implementation. *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI)*.

Sondik, E. (1978). The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research, 26*, 282–304.

Spaan, M., & Vlassis, N. (2005). Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research, 24*, 195–220.

Spaan, M. T., & Vlassis, N. (2004). A point-based POMDP algorithm for robot planning. *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 2399–2404).

Stone, P., Sutton, R. S., & Kuhlmann, G. (2005). Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior, 13*, 165–188.

Strehl, A. L., & Littman, M. L. (2005). A theoretical analysis of model-based interval estimation. *ICML '05: Proceedings of the 22nd international conference on Machine learning* (pp. 856–863). New York, NY, USA: ACM Press.

Strens, M. (2000). A Bayesian framework for reinforcement learning. *Proceedings ICML*.

Strens, M., & Moore, A. (2002). Policy search using paired comparisons. *JMLR, 3*, 921–950.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. MIT Press.

Sutton, R. S. (1984). *Temporal credit assignment in reinforcement learning*. Doctoral dissertation, University of Massachusetts, Amherst, MA 01003.

Sutton, R. S. (1991). Planning by incremental dynamic programming. *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 353–357). Morgan Kaufmann.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems* (pp. 1038–1044).

Tesauro, G. (1994). Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation, 6*, 215–219.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika, 25*, 285–294.

Thompson, W. R. (1935). On the theory of apportionment. *Amer. J. Math, 57*, 450–456.

Thrun, S. (2000). Monte Carlo POMDPs. *Advances in Neural Information Processing Systems (NIPS 12)*.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT Press.

Thrun, S., & Schwartz, A. (1993). Issues in Using Function Approximation for Reinforcement Learning. *Proceedings of the 1993 Connectionist Models Summer School*. Hillsdale, NJ: Lawrence Erlbaum.

Toh, K., Todd, M., & Tutuncu, R. (1999). SDPT3–a Matlab software package for semidefinite programming. *Optimization Methods and Software, 11*.

Tsitsiklis, J., & Roy, B. V. (1999). Average cost temporal-difference learning. *Automatica, 35*, 1799–1808.

Tsitsiklis, J., & Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control, 42*, 674–690.

Tsitsiklis, J. N. (1994a). Asynchronous stochastic approximation and q-learning. *Machine Learning., 16*, 185–202.

Tsitsiklis, J. N. (1994b). A short proof of the Gittins index theorem. *The Annals of Applied Probability, 4*, 194–199.

Tsitsiklis, J. N., & Roy, B. V. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control, 42*, 674–690.

Tsitsiklis, J. N., & Roy, B. V. (2002). On average versus discounted reward temporal-difference learning. *Machine Learning, 49*, 179–191.

Varaiya, P. P., Warland, J. C., & Buyukkoc, C. (1985). Extensions of the multiarmed bandit problem: The discounted case. *IEEE Trans. on Auto. Control, AC-30*, 426–439.

Wang, T., Bowling, M., & Schuurmans, D. (2007). Dual representations for dynamic programming and reinforcement learning. *Proceedings of the 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning* (pp. 44–51). (Winner of the Best Student Paper Award).

Wang, T., Lizotte, D., Bowling, M., & Schuurmans, D. (2005). Bayesian sparse sampling for on-line reward optimization. *Proceedings of the Twenty-second International Conference on Machine Learning (ICML)* (pp. 961–968).

Wang, T., Poupart, P., Bowling, M., & Schuurmans, D. (2006). Compact, convex upper bound iteration for approximate POMDP planning. *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI)* (pp. 1245–1251).

Watkins, C. (1989). *Learning from delayed rewards*. Doctoral dissertation, King's College Cambridge.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*, 279–292.

Wiering, M. (1999). *Explorations in efficient reinforcement learning*. Doctoral dissertation, University Amsterdam.

Williams, C. (1999). Prediction with Gaussian processes. In M. Jordan (Ed.), *Learning in graphical models*. MIT Press.

Wyatt, J. (1997). *Exploration and inference in learning from reinforcement*. Doctoral dissertation, University of Edinburgh.

Wyatt, J. (2001). Exploration control in reinforcement learning using optimistic model selection. *Proceedings ICML*.

Xu, X., gen He, H., & Hu, D. (2002). Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Intelligence Research, 16*, 259–292.

Zhang, N., & Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research, 14*, 29–51.

Zhou, R., & Hansen, E. (2001). An improved grid-based approximation algorithm for POMDPs. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*.

Appendix A

Notation for Chapter 3

S	set of states	$\|S\|$	$s \in S$
A	set of actions	$\|A\|$	$a \in A$
$\mathbf{r}$	reward model	$\|S\|\|A\| \times 1$	$\mathbf{r}_{(sa)}$
π	stationary policy	$\|S\|\|A\| \times 1$	$\pi_{(sa)}$
P	transition model	$\|S\|\|A\| \times \|S\|$	$P_{(sa,s')}$
Π	sparse policy matrix	$\|S\| \times \|S\|\|A\|$	$\Pi_{(s,s'a)}$
Z	stationary state-action distribution	$\|S\|\|A\| \times \|S\|\|A\|$	
$\mathbf{v}$	state value function	$\|S\| \times 1$	$\mathbf{v}_{(s)}$
$\mathbf{q}$	state-action value function	$\|S\|\|A\| \times 1$	$\mathbf{q}_{(sa)}$
$\boldsymbol{\mu}$	initial state distribution	$\|S\| \times 1$	
$\boldsymbol{\nu}$	initial state-action distribution	$\|S\|\|A\| \times 1$	
Ξ	marginalization matrix	$\|S\| \times \|S\|\|A\|$	
$\mathbf{c}$	joint prob. distr. over state pairs	$\|S\| \times 1$	$\mathbf{c}_{(s)}$
$\mathbf{d}$	joint prob. distr. over state-action pairs	$\|S\|\|A\| \times 1$	$\mathbf{d}_{(sa)}$
M	prob. of discounted state visits	$\|S\| \times \|S\|$	$M_{(s,s')}$
H	prob. of discounted state-action visits	$\|S\|\|A\| \times \|S\|\|A\|$	$H_{(sa,s'a')}$
$\mathbf{w}$	adjustable weights	$k \times 1$	
Φ	bases for the primal representation	$\|S\|\|A\| \times k$	
Ψ	bases for the dual representation	$(\|S\|\|A\|)^2 \times k$	
Υ	basis matrix, nonnegative, *row* normalized	$\|S\|\|A\| \times \|S\|\|A\|$	
Γ	$(\mathbf{r}^\top \otimes I)\Psi$	$\|S\|\|A\| \times k$	

Appendix B

Tabular Reinforcement Learning

Beyond demonstrating novel dual representations for planning with a given MDP model, I can also show how these representations can be used to derive novel forms of learning algorithms without a MDP model as well. In reinforcement learning algorithms, e.g., temporal difference (TD) evaluation, Sarsa, and Q-learning, the environmental model P and $\mathbf{r}$ are unknown, but instead access to the environment is limited to the selection of actions and observation of state transitions and rewards.

B.1 Temporal Difference Evaluation

First, I address TD prediction methods for policy evaluation. In the primal case, the value of a given policy can be estimated by the standard TD evaluation algorithm, with the update step given by

$$\mathcal{O}\mathbf{v}_{(s)} = (1-\alpha)\mathbf{v}_{(s)} + \alpha\left[r + \gamma\mathbf{v}_{(s')}\right] \tag{B.1}$$

Note that a simple entry of the state value vector is updated. The computational complexity of primal TD evaluation algorithm is $O(1)$.

In the dual representation, an analogous TD evaluation algorithm can be derived with respect to the state distribution matrix. In this case, the update step is given by

$$\mathcal{O}M_{(s,:)} = (1-\alpha)M_{(s,:)} + \alpha\left[(1-\gamma)\mathbf{e}_s^\top + \gamma M_{(s',:)}\right] \tag{B.2}$$

and $\mathbf{e}_s$ is the vector of all zeros except for a 1 in the s^{th} position. Note that a row of the discounted state visit distribution matrix is updated, and no reference to the immediate reward $\mathbf{r}$ is made. The computational complexity of dual TD evaluation

algorithm is $O(|S|)$, which is more expensive than the primal TD algorithm unless the sparsity of the matrix M can be exploit.

B.2 Sarsa: On-Policy TD Control

Next we consider the on-policy control problem. In the primal case, the Sarsa algorithm approximates the action-value function of the policy being followed, and interleaves this with policy improvement. The action-value update step is

$$\mathcal{O}\mathbf{q}_{(sa)} = (1-\alpha)\mathbf{q}_{(sa)} + \alpha\left[r + \gamma\mathbf{q}_{(s'a')}\right] \tag{B.3}$$

The policy improvement step here is implicit since the next action is chosen greedily with probability $1-\epsilon$ with respect to the estimated update $\mathbf{q}$. That is, choosing $a' = \text{argmax}_{a'}\, \mathbf{q}_{(s'a')}$ from s' with probability $1-\epsilon$.

In the dual representation, an analogous Sarsa algorithm can be derived with respect to the state-action distribution matrix. In this case, the distribution update can be given by

$$\mathcal{O}H_{(sa,:)} = (1-\alpha)H_{(sa,:)} + \alpha\left[(1-\gamma)\mathbf{e}_{sa}^{\top} + \gamma H_{(s'a',:)}\right] \tag{B.4}$$

and $\mathbf{e}_{sa}$ is the vector of all zeros except for a 1 in the sa^{th} position. Note that here a row of the state-action distribution matrix is updated, and again, no reference to the immediate reward $\mathbf{r}$ is made.

The computational complexity of primal Sarsa algorithm is $O(1)$ per time step while the computational complexity of dual Sarsa algorithm is $O(|S||A|)$. The dual Sarsa algorithm can be much more expensive than standard Sarsa algorithm unless the sparsity of the matrix H can be exploited.

B.3 Q-Learning: Off-Policy TD Control

Finally, I consider the off-policy control problem. In the primal case, the Q-learning algorithm directly approximates $\mathbf{q}^*$, the optimal action-value function. Here the state-action value update is

$$\mathcal{M}\mathbf{q}_{(sa)} = (1-\alpha)\mathbf{q}_{(sa)} + \alpha\left[r + \gamma \max_{a'} \mathbf{q}_{(s'a')}\right] \tag{B.5}$$

The update is a convex combination of the new estimate and old estimate.

In the dual representation, an analogous Q-Learning algorithm can be derived with respect to the state-action distribution matrix, by using the distribution update

$$\mathcal{M}H_{(sa,:)} = (1-\alpha)H_{(sa,:)} + \alpha\left[(1-\gamma)\mathbf{e}_{sa}^{\top} + \gamma H_{(s'a'^{*},:)}\right] \quad \text{(B.6)}$$

where $a'^{*} = \text{argmax}_{a'} H_{(s'a',:)}\mathbf{r}$ and $\mathbf{e}_{(sa)}$ is the vector of all zeros except for a 1 in the sa^{th} position.

The computational complexity of primal Q-learning algorithm is $O(|A|)$ per time step while the computational complexity of dual Q-learning algorithm is $O(|S||A|^2)$. The dual Q-learning algorithm can be much more expensive than standard Q-learning algorithm unless the sparsity of the matrix H can be exploited.

Appendix C

Approximate Reinforcement Learning

In a reinforcement learning scenario, the gradient update operators are applied pointwise with a single sampled transition, in place of a full expectation with respect to $P\Pi$ in dynamic programming algorithms. This update is called sampled gradient operator.

Primal Case

In the primal, recall that the gradient of objective $J_{\mathbf{q}} = \frac{1}{2}\|\mathbf{q} - \hat{\mathbf{q}}\|_{\mathbf{z}}^2 = \frac{1}{2}\|\mathbf{q} - \Phi\mathbf{w}\|_{\mathbf{z}}^2$ with respect to $\mathbf{w}$ is

$$\begin{aligned} \nabla_{\mathbf{w}} J_{\mathbf{q}} &= \Phi^{\top} Z(\hat{\mathbf{q}} - \mathbf{q}) \\ &= \sum_{sa} \Phi_{(sa,:)}^{\top} Z_{(sa,sa)}(\hat{\mathbf{q}}_{(sa)} - \mathbf{q}_{(sa)}) \\ &= \mathrm{E}_{(sa)\sim Z} \Phi_{(sa,:)}^{\top}(\hat{\mathbf{q}}_{(sa)} - \mathbf{q}_{(sa)}) \end{aligned} \tag{C.1}$$

where $diag(Z)$ corresponds to the stationary distribution over state-action pairs. Sampling the expectation in the above equation, we can get the sampled gradient

$$\widetilde{\nabla_{\mathbf{w}} J_{\mathbf{q}}} = \Phi_{(sa,:)}^{\top}(\hat{\mathbf{q}}_{(sa)} - \mathbf{q}_{(sa)}) \tag{C.2}$$

Therefore, we have the stochastic gradient update of the weight vector

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - \alpha \widetilde{\nabla_{\mathbf{w}} J_{\mathbf{q}}} \\ &= \mathbf{w}_t - \alpha \Phi_{(sa,:)}^{\top}(\hat{\mathbf{q}}_{(sa)} - \mathbf{q}_{(sa)}) \end{aligned} \tag{C.3}$$

Since $\hat{\mathbf{q}} = \Phi\mathbf{w}$, we have $\hat{\mathbf{q}}_{(sa)} = \Phi_{(sa,:)}\mathbf{w}$. Plugging it back into the above update

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\Phi_{(sa,:)}^\top(\Phi_{(sa,:)}\mathbf{w}_t - \mathbf{q}_{(sa)}) \tag{C.4}$$

Because the true value of $\mathbf{q}$ is unknown, we approximate it by sampling either $\mathcal{O}\hat{\mathbf{q}} = \mathbf{r} + \gamma P\Pi\hat{\mathbf{q}}$ or $\mathcal{M}\hat{\mathbf{q}} = \mathbf{r} + \gamma P\Pi^*[\hat{\mathbf{q}}]$. This yields the stochastic gradient updates

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\Phi_{(sa,:)}^\top(\Phi_{(sa,:)}\mathbf{w}_t - \mathbf{r}_{(sa)} - \gamma\Phi_{(s'a',:)}\mathbf{w}_t) \tag{C.5}$$

and

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\Phi_{(sa,:)}^\top(\Phi_{(sa,:)}\mathbf{w}_t - \mathbf{r}_{(sa)} - \gamma\max_{a'}\Phi_{(s'a',:)}\mathbf{w}_t) \tag{C.6}$$

respectively for the on-policy and off-policy cases.

Thus, the unknown parameter $\mathbf{w}$ can be learned iteratively with the above updates. Then one can recover the estimation of action value from $\hat{\mathbf{q}} = \Phi\mathbf{w}$.

The computational complexity for one iteration of sampled $\mathcal{GO}$ (see Equation C.5) is k, which is the number of bases in the linear approximation. The computational complexity for one iteration of sampled $\mathcal{GM}$ (see Equation C.6) is $k|A|$. The total computational complexity of the primal sampled gradient algorithm is the product of the number of iterations (samples) and the cost for each iteration.

Dual Case

In the dual, recall that the gradient of objective $J_H = \frac{1}{2}\|H - \hat{H}\|_{\mathbf{z},\mathbf{r}}^2$ with respect to $\mathbf{w}$ is

$$\begin{aligned}
\nabla_{\mathbf{w}}J_H &= \Gamma^\top Z(\mathbf{r}^\top \otimes I)(\hat{h} - h) \\
&= \Gamma^\top Z\left((\mathbf{r}^\top \otimes I)\Psi\mathbf{w} - (\mathbf{r}^\top \otimes I)h\right) \\
&= \Gamma^\top Z(\Gamma\mathbf{w} - H\mathbf{r}) \\
&= \sum_{sa}\Gamma_{(sa,:)}^\top Z_{(sa,sa)}(\Gamma_{(sa,:)}\mathbf{w} - H_{(sa,:)}\mathbf{r}) \\
&= \mathrm{E}_{(sa)\sim Z}\Gamma_{(sa,:)}^\top(\Gamma_{(sa,:)}\mathbf{w} - H_{(sa,:)}\mathbf{r})
\end{aligned} \tag{C.7}$$

Sampling the expectation in the above equation, we can get the sampled gradient

$$\widetilde{\nabla_{\mathbf{w}}J_H} = \Gamma_{(sa,:)}^\top(\Gamma_{(sa,:)}\mathbf{w} - H_{(sa,:)}\mathbf{r}) \tag{C.8}$$

Therefore, we have the stochastic gradient update of the weight vector

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - \alpha\widetilde{\nabla_{\mathbf{w}} J_H} \\ &= \mathbf{w}_t - \alpha\Gamma_{(sa,:)}^{\top}(\Gamma_{(sa,:)}\mathbf{w}_t - H_{(sa,:)}\mathbf{r}) \end{aligned} \tag{C.9}$$

Because the true value of H is unknown, we approximate it by sampling either $\mathcal{O}\hat{H} = (1-\gamma)I + \gamma P\Pi\hat{H}$ or $\mathcal{M}\hat{H} = (1-\gamma)I + \gamma P\Pi_r{}^*[\hat{H}]$. This yields the stochastic gradient updates

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\Gamma_{(sa,:)}^{\top}\left(\Gamma_{(sa,:)}\mathbf{w}_t - (1-\gamma)\mathbf{r}_{(s'a')} - \gamma\Gamma_{(s'a',:)}\mathbf{w}_t\right) \tag{C.10}$$

and

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\Gamma_{(sa,:)}^{\top}\left(\Gamma_{(sa,:)}\mathbf{w}_t - (1-\gamma)\mathbf{r}_{(s'a')} - \max_{a'}\gamma\Gamma_{(s'a',:)}\mathbf{w}_t\right) \tag{C.11}$$

for the on-policy case and off-policy cases respectively.

Thus, the unknown parameter w can be learned iteratively with the above updates. Note that we need check if the w satisfies the constraints of the weights (positive and normalized) and enforce constraints if needed. Then one can recover the estimation of state-action visit distribution from $\hat{H} = reshape(\Psi\mathbf{w})$.

The computational complexity for one iteration of sampled $\mathcal{GO}$ (see Equation C.10) is k, which is the number of bases in the linear approximation. The computational complexity for one iteration of sampled $\mathcal{GM}$ (see Equation C.11) is $k|A|$. The cost of dual stochastic gradient updates are the same as the primal ones. However, in the dual one needs to check if the learned w satisfies the constraints. If not, an extra step of enforcing the constraints of the weights is required, e.g., projecting w to its linear constraints $\mathbf{w}^T\mathbf{1} = 1$ costs $O(k^2)$. The total computational complexity of the dual sampled gradient algorithm is the product of the number of iterations (samples) and the cost for each iteration (update and enforcing constraints).

Although approximate dual algorithms are interesting, they can be computationally more expensive than their counterparts in the primal. The theoretical analysis of the convergence of the approximate dual reinforcement learning algorithms has not been conducted yet.

Other Algorithms

Here I only give a brief review of primal approximate reinforcement learning algorithms.

It is proven that TD(λ) with linear function approximation converges (with probability one) in both the discounted (Tsitsiklis & Roy, 1997) and average reward case (Tsitsiklis & Roy, 1999; Tsitsiklis & Roy, 2002). TD(λ) can diverge with nonlinear function approximation (Tsitsiklis & Roy, 1997).

Least-squares temporal difference (LSTD) learning algorithm (Bradtke & Barto, 1996; Boyan, 1999) is an approximate policy evaluation method, which minimizes mean squared TD errors over sampled data. The state value function is represented as a linear combination of features with unknown parameters. LSTD first estimates its model (**A** and **b**) from sampled data, then it computes the unknown parameters from the estimated model. The underlying update of LSTD is using projection operator $\mathcal{P}$ on approximate state value function. Recursive LSTD (Xu et al., 2002) and iLSTD (Geramifard et al., 2006) are approaches to speed up LSTD.

Least-squares policy iteration (LSPI) is approximate policy iteration, which is a combination of policy evaluation LSTDQ (LSTD on state-action value function) and policy improvement. It is a stable algorithm and requires the storage of samples to estimate the coefficients of a linear system (matrix $\tilde{\mathbf{A}}$ and vector $\tilde{b}$) to compute the learned parameters **w** (Lagoudakis & Parr, 2003). An approximate policy iteration algorithm was shown to be able to find an "approximately" optimal solution in polynomial time (Kakade & Langford, 2002). Their algorithm assumes that a restart distribution over states is given so that the trajectories can be sampled from the same state during estimation and treats the step of generating a greedy policy as a regression problem.

Approximate Sarsa is a combination of state-action value update $\mathcal{GO}$ (see Equation C.5) and an implicit policy improvement. It has been shown that Sarsa(0) with linear function approximation cannot diverge, but might oscillate (Gordon, 2000; Singh et al., 2000).

Q-learning with linear function approximation whose update is $\mathcal{GM}$ (see Equation C.6), is a common algorithm attempted in reinforcement learning. It is inter-

esting as explained in Chapter 1. However, it can diverge (Bradtke, 1993; Boyan & Moore, 1995; Baird, 1995). Fortunately, an off-policy algorithm that combines TD(λ) over state-action pairs with importance sampling is proven to converge with probability one (Precup et al., 2001).

Appendix D

Common Probability Distributions

Distribution	Probability $P(x)$	E$[X]$	$Variance(X)$
$Bernoulli(x \mid \theta)$ $0 < \theta < 1$	$\theta^x(1-\theta)^{1-x}, x = 0, 1$	θ	$\theta(1-\theta)$
$Binomial(x \mid \theta, t)$ $0 < \theta < 1, t = 1, 2, ...$	$\binom{t}{x}\theta^x(1-\theta)^{t-x}, x = 0, 1, .., t$	$t\theta$	$t\theta(1-\theta)$
$Gaussian(x \mid \mu, \sigma^2)$	$\frac{1}{\sqrt{2\pi}\sigma}e^{-(x-\mu)^2/2\sigma^2}, -\infty < x < +\infty$	μ	σ^2
$Beta(x \mid \alpha, \beta), \alpha, \beta > 0$	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}, 0 < x < 1$	$\frac{\alpha}{\alpha+\beta}$	$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$
$\Gamma(x \mid \alpha, \beta), \alpha, \beta > 0$	$\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}, x > 0$	$\alpha\beta^{-1}$	$\alpha\beta^{-2}$

Table D.1: Some common probability distributions. The quantity x is a realization of the random variable, and $\theta, \mu, \sigma, \alpha, \beta$ are unknown parameters of the distributions.

Appendix E

Some Useful Conjugate Priors

Likelihood $P(x \mid \theta)$	Conjugate Prior $P(\theta)$	Posterior $P(\theta \mid x)$
$Binomial(x \mid \theta, t)$	$Beta(\theta \mid \alpha, \beta)$	$Beta(\theta \mid \alpha + s, \beta + t - s)$
$Poisson(x \mid \lambda)$	$Gamma(\lambda \mid \alpha, \beta)$	$Gamma(\lambda \mid \alpha + t, \beta + 1)$
$Gaussian(x \mid \mu, \sigma^2)$	$Gaussian(\mu \mid \mu_0, \sigma_0^2)$	$Gaussian(\mu \mid \mu_1, \sigma_1^2)$

Table E.1: Some useful conjugate priors. The quantity x is a realization of the random variable, and $\theta, \mu, \sigma, \alpha, \beta$ are unknown parameters of the distributions. Here the notation x stands for the observed value, and θ is the generic symbol for the parameter to infer (corresponding to μ and σ of a Gaussian, θ of a binomial and λ of a Poisson distribution). The quantity t is the total number of observations, the notation s represents the number of outcome $1's$ (successes).

Appendix F

Other Related Work

F.1 Exponential Family

In probability and statistics, the exponential family is an important class of probability distributions as they have nice algebraic properties that make them mathematically convenient. All members of the *exponential family* have conjugate priors and can express a wide range of beliefs. The Bernoulli, Gaussian, binomial, multinomial, Gamma, Poisson and Beta distributions are members of the exponential family and their probability distribution functions can be expressed as a general form

$$P(x \mid \eta) = h(x) \exp\left\{\eta^{\top} T(x) - A(\eta)\right\} \tag{F.1}$$

where

- x is a random variable,
- η is the vector of natural parameters whose transpose $\eta^{\top} = (\eta_1, ..., \eta_n)$ is a row vector,
- $h(x)$ reflects the underlying measure with respect to which $p(x|\eta)$ is a probability density distribution,
- $T(x)$ is a sufficient statistic[1] of the distribution, which is a column vector whose number of scalar components is the same as that of η so that $\eta^{\top} T(x)$ is a scalar,
- $A(\eta)$ is a normalization factor.

[1]Note that the concept of sufficient statistic applies more broadly than just to members of the exponential family.

The Bernoulli Distribution

A Bernoulli experiment has only two possible outcomes: 1 (*success*) with probability θ and 0 (*failure*) with probability $1-\theta$. Let a random variable X represent the success of an experiment. The probability mass function of a Bernoulli random variable X is given as follows.

$$P(x \mid \theta) = \theta^x (1-\theta)^{1-x} = \exp\left\{ \log\left(\frac{\theta}{1-\theta}\right) x + \log(1-\theta) \right\} \tag{F.2}$$

The Bernoulli distribution is an exponential family distribution with

$$\begin{aligned} \eta &= \frac{\theta}{1-\theta} \\ T(x) &= x \\ A(\eta) &= -\log(1-\theta) = \log(1+e^{\eta}) \\ h(x) &= \delta_0(x) + \delta_1(x) \end{aligned}$$

where $\delta_y(x)$ is the unit impulse function, such that $\int_x f(x)\delta_y(x)dx = f(y)$ for all continuous f. Note that the relationship between η and θ is invertible; therefore we have $\theta = \frac{1}{1+e^{-\eta}}$ by solving the above equations.

Beta Posterior Update. A conjugate prior for the Bernoulli parameter θ is a Beta distribution. That is, consider a Bernoulli process with unknown parameter θ, and choose a Beta distribution $Beta(\alpha, \beta)$ to be the prior over θ, given by $P(\theta|\alpha,\beta) = \theta^{\alpha-1}(1-\theta)^{\beta-1}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$. Here Γ is the Gamma function. The parameter α denotes the "prior" number of *successes* observed; the parameter β denotes the "prior" number of *failures* observed before the new observations.[2] After observing the outcome of the Bernoulli process, either *success* or *failure*, we can update the Beta posterior simply by adding one to the corresponding Beta parameter.

$$\textit{Beta posterior} \quad := \quad \begin{cases} \alpha := \alpha + 1 & \text{if } \textit{success} \text{ is observed} \\ \beta := \beta + 1 & \text{if } \textit{failure} \text{ is observed} \end{cases} \tag{F.3}$$

That is, the posterior density over θ is given by $P(\theta \mid s,t) = Beta(\theta \mid \alpha+s, \beta+t-s)$, where s the number of successes and $t-s$ is the number of failures.

[2] $Beta(1,1)$ is the uniform distribution.

The Univariate Gaussian Distribution

The univariate Gaussian (normal) density of a random variable X with mean μ and variance σ^2 can be written as follows:

$$\mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-(x-\mu)^2/2\sigma^2\right\} = \frac{1}{\sqrt{2\pi}} \exp\left\{\frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2 - \ln\sigma\right\} \tag{F.4}$$

It is in the exponential family with

$$\eta = \begin{bmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{bmatrix}$$
$$T(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$$
$$A(\eta) = \frac{\mu}{2\sigma^2} + \ln\sigma$$
$$h(x) = \frac{1}{\sqrt{2\pi}}$$

Univariate Gaussian Posterior Update. A conjugate prior for the Gaussian mean μ is just another Gaussian. Consider a data-generating process that can be described by the likelihood $P(x|\mu) = \mathcal{N}(x \mid \mu, \sigma^2)$, where parameter μ is unknown and parameter σ is known. Assume we have prior knowledge about the distribution of μ of the form $P(\mu) = \mathcal{N}(\mu \mid \mu_0, \sigma_0^2)$, where both μ_0 is our best guess of μ and σ_0^2 is our uncertainty about the guess. After observing t independent samples, $x_1, ..., x_t$, we have the posterior distribution

$$P(\mu \mid x_1, x_2, ..., x_t) = \mathcal{N}(\mu \mid \tilde{\mu}, \tilde{\sigma}^2) \tag{F.5}$$

where

$$\tilde{\mu} = \frac{t/\sigma^2}{t/\sigma^2 + 1/\sigma_0^2}\bar{x} + \frac{1/\sigma_0^2}{t/\sigma^2 + 1/\sigma_0^2}\mu_0$$
$$\tilde{\sigma}^2 = (t/\sigma^2 + 1/\sigma_0^2)^{-1}$$

and $\bar{x} = \frac{1}{t}\sum_{k=1}^{t} x_k$ is the sample mean (Duda et al., 2001).

The Multivariate Gaussian Distribution

The multivariate Gaussian (normal) distribution of a random vector $\mathbf{X} \in \mathbb{R}^n$ with

mean $\boldsymbol{\mu}$ (an $n \times 1$ vector), and variance $\boldsymbol{\Sigma}$ (an $n \times n$ symmetric positive definite matrix) can be written as follows:

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{F.6}$$

Multivariate Gaussian Posterior Update. A conjugate prior for the multivariate Gaussian mean $\boldsymbol{\mu}$ is also just another multivariate Gaussian (Duda et al., 2001). Consider a data-generating process that can be described by the likelihood $P(\mathbf{x}|\boldsymbol{\mu}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where the parameter vector $\boldsymbol{\mu}$ is unknown and the parameter vector $\boldsymbol{\Sigma}$ is known. Assume prior knowledge about the distribution of the unknown vector $\boldsymbol{\mu}$ in the form $P(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu} \mid \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. After observing t independent samples, $\mathbf{x}_1, ..., \mathbf{x}_t$, we have the posterior distribution

$$P(\boldsymbol{\mu} \mid \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t) = \mathcal{N}(\boldsymbol{\mu} \mid \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \tag{F.7}$$

where

$$\tilde{\boldsymbol{\mu}} = \boldsymbol{\Sigma}_0\left(\boldsymbol{\Sigma}_0 + \frac{1}{t}\boldsymbol{\Sigma}\right)^{-1}\bar{\mathbf{x}} + \frac{1}{t}\boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}_0 + \frac{1}{t}\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu}_0$$

$$\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_0\left(\boldsymbol{\Sigma}_0 + \frac{1}{t}\boldsymbol{\Sigma}\right)^{-1}\frac{1}{t}\boldsymbol{\Sigma}$$

and $\bar{\mathbf{x}} = \frac{1}{t}\sum_{k=1}^{t}\mathbf{x}_k$ is the sample mean (an $n \times 1$ vector).

Gaussian Processes

A Gaussian process can be thought of as a possibly infinite dimensional generalization of the multivariate Gaussian. Formally, a *Gaussian process* $\{F(\mathbf{x}) \mid \mathbf{x} \in X\}$ is a (finite, countably infinite, or uncountably infinite) collection of random variables indexed by a variable or vector $\mathbf{x} \in X$ such that any finite subset of the F-variables has a joint multivariate Gaussian distribution (Williams, 1999). A Gaussian Process can be fully specified by its mean function $\mu(\mathbf{x})$ and covariance function $K(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$, $\mathbf{x}' \in X$, and K is a symmetric and positive-definite function (*kernel*). For any finite subset of indices $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t \in X$, we have

$F_t \sim \mathcal{N}(\boldsymbol{\mu}_t, K_t)$. That is

$$\begin{bmatrix} F(\mathbf{x}_1) \\ F(\mathbf{x}_2) \\ \vdots \\ F(\mathbf{x}_t) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{x}_1) \\ \mu(\mathbf{x}_2) \\ \vdots \\ \mu(\mathbf{x}_t) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_t) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_t) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}_1) & k(\mathbf{x}_t, \mathbf{x}_2) & \cdots & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix} \right) \tag{F.8}$$

Gaussian processes allow us to express priors over functions. One property of Gaussian processes is that the posterior distribution is Gaussian given that both the prior distribution and likelihood are Gaussian. As with other conjugate distributions, this is computationally convenient.

Gaussian Process Posterior Update. Consider a data-generating process that can be described by a Gaussian process $\{F(\mathbf{x}) \mid \mathbf{x} \in X\}$. Normally in the posterior update for Gaussian processes we do not assume that we observe F-values directly, but get noisy observations corrupted by some additional independent Gaussian noise. That is, we define

$$Y(\mathbf{x}) = F(\mathbf{x}) + N(\mathbf{x}) \tag{F.9}$$

where F is the underlying Gaussian process and N is an independent zero mean Gaussian process. Then given a sequence of samples $(\mathbf{x}_i, y_i)$, $i = 1, ..., t$, we have t equations

$$Y_t = F_t + N_t \tag{F.10}$$

where $Y_t = [Y(\mathbf{x}_1), ..., Y(\mathbf{x}_t)]^\top$ and $N_t = [N(\mathbf{x}_1), ..., N(\mathbf{x}_t)]^\top$. From Equation F.8 and the definition of N we then have

$$F_t \sim \mathcal{N}(\boldsymbol{\mu}_t, K_t), \quad N_t \sim \mathcal{N}(0, \boldsymbol{\Sigma}_t) \tag{F.11}$$

where F_t and N_t are independent. Therefore, from Equation F.10 we get

$$Y_t \sim \mathcal{N}(\boldsymbol{\mu}_t, K_t + \boldsymbol{\Sigma}_t) \tag{F.12}$$

The posterior $F(\mathbf{x})$ conditioned on the observations Y_t is Gaussian distributed. The mean $\mathrm{E}[F(\mathbf{x}) \mid Y_t]$ and the covariance $\mathbf{Cov}(F(\mathbf{x}), F(\mathbf{x}') \mid Y_t)$ can be updated by

$$\mathrm{E}[F(\mathbf{x}) \mid Y_t = \mathbf{y}_t] = \boldsymbol{\mu}_0(\mathbf{x}) + \mathbf{k}_t(\mathbf{x})^\top (K_t + \boldsymbol{\Sigma}_t)^{-1} \mathbf{y}_t \tag{F.13}$$

$$\mathrm{Cov}(F(\mathbf{x}), F(\mathbf{x}') \mid Y_t = \mathbf{y}_t) = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_t(\mathbf{x})^\top (K_t + \Sigma_t)^{-1} \mathbf{k}_t(\mathbf{x}') \quad \text{(F.14)}$$

where $\boldsymbol{\mu}_0 = [\mu_0(\mathbf{x}_1), ..., \mu_0(\mathbf{x}_t)]^\top$, $\mathbf{y}_t = [y_1, ..., y_t]^\top$, $\mathbf{k}_t(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}), ..., k(\mathbf{x}_t, \mathbf{x})]^\top$, and K_t is the same as in Equation F.8.

F.2 Multi-armed Bandit Problem

The multi-armed bandit problem is a mathematical model for the problem of optimizing the total reward accumulated by choosing to play a number of arms ("slot machines"), whose reward distributions are unknown and independent.

Early work on the multi-armed bandit problem (Berry & Fristedt, 1985) concentrated on finite horizon problems with Bernoulli arms. These were introduced in the framework of clinical trials by Thompson (1933) for two treatments whose outcomes are either 1 (*success*) or 0 (*failure*) and the prior distribution over the outcomes is uniform. Robbins (1952) reintroduced the multi-armed bandit problem from a non-Bayesian viewpoint, proposed the play-the-winner/switch-from-a-loser strategy, and discussed the asymptotic behavior of the minimax decision rules for the Bernoulli case. Bradt, Johnson and Karlin (1956) wrote the first paper to search for Bayes decision rules for this problem. Bellman (1956) adopted a Bayesian formulation of multi-armed bandit for the infinite-horizon discounted case.

There are various multi-armed bandit problems, which can be categorized from dimensions such as discrete-time vs. continuous-time bandits, finite horizon vs. infinite horizon case, and discount factor (uniform, geometric, and exponential).

Here we consider the basic version of the bandit problem (Karoui & Karatzas, 1993) to introduce the key terms. Given a collection of N statistically independent reward processes, a decision maker chooses one process to "activate" at each decision stage t. The activated process yields an immediate reward and changes its state; the other processes remain "frozen" in their current states and yield no reward. The goal for the decision maker is to choose a sequence of processes to maximize the expected total discounted rewards

$$E\left[\sum_{t=0}^{\infty} \gamma^t R^{a_t}(S_t^{a_t})\right] \quad \text{(F.15)}$$

An optimal strategy for a multi-armed bandit problem can be determined by computing the Gittins index of the state of each arm, then choosing the arm with highest Gittins index (Gittins, 1989). Consider a single arm. The Gittins index of state $s_t = i$ of the arm, $g(i)$, is defined as the supremum of expected discounted reward per expected unit of discounted time over all possible stopping rules

$$g(i) = \sup_{\tau>0} \frac{E\left[\sum_{t=0}^{\tau-1} \gamma^t R(s_t)|s_0 = i\right]}{E\left[\sum_{t=0}^{\tau-1} \gamma^t|s_0 = i\right]} \tag{F.16}$$

Theorem 9 *The N-armed bandit problem (with independent arms and geometric discount) can be solved by solving N one-armed bandit problems.*[3]

Theorem 10 *The optimal policy is to play the bandit arm with the greatest Gittins index at each stage.*

Rigorous proofs of the above two theorems could be found in (Tsitsiklis, 1994b; Bertsimas & Nino-Mora., 1996; Frostig & Weiss, 1999).

Remark 11 *The multi-armed bandit problem can be decomposed into simpler, low-dimensional subproblems (stopping problems). Therefore, the problem of finding optimal policies for the original MDP (Markov Decision Process) can be reduced to a sequence of stopping problems (Varaiya et al., 1985; Katehakis & Veinott, 1987).*

The papers (Auer et al., 1995; Mickova, 2000; Kleinberg, 2005) provide some recent work on the multi-armed bandit problem.

F.3 Batch Methods for Action Selection

Although batch (off-line) learning is a slightly unnatural model for reinforcement learning, important theoretical results have been obtained which show that near optimal policies can be learned in time polynomial in the size of the state and action spaces (as well as other variables). Even though I did not necessarily work on this topic for my thesis, for completeness, I include a brief discussion here.

[3]The one-armed bandit problem can be viewed as two-armed bandit problem. The reward distribution of arm 1 is unknown while the arm 2 gives a constant return each time when it is pulled.

Explicit Exploration Exploitation. The Explicit Exploration Exploitation (E^3) algorithm (Kearns & Singh, 1998) was considered the first provably optimal polynomial time algorithm for optimizing average reward on an MDP. It divides the states of the MDP being learned into two parts: the known states and unknown states. A good estimate of an unknown state transition probability function can be gained by visiting the unknown state a sufficient number of times. When in a known state of the MDP, planning is used to determine a policy that can efficiently reach an unknown state, which is necessary to reduce the number of steps required to visit some unknown state. The algorithm can be shown to produce a close approximation to the optimal value function with high probability in polynomial time.

R-max. As a follow up to this line of research, the R-max algorithm (Brafman & Tennenholtz, 2001) employs a much simpler idea but also achieves polynomial time guarantees. R-max puts high artificial rewards on unknown state-action pairs (optimistic initialization) and then simply executes exploitive actions by planning in the optimistic MDP. After observing a sufficient number of transitions from a particular state and action, the new optimistic MDP is solved for a new policy. Similar results to E^3 can be established, but for a much simpler algorithm.

Both methods are theoretically interesting as they proved that a near optimal policy could be found in polynomial time with high probability. However, they do not attempt to maximize the rewards obtained during the learning. In addition, the number of actions taken is chosen (up to a polynomial bound) by the methods themselves and not under direct external control.